Autonomous University of Madrid

Master Thesis

On the possibility to find coordinates by random features

Author:
Carlos Asensio Pizarro

Supervisor:
Dr. Kostadin Koroutchev

A thesis submitted in fulfilment of the requirements for the degree of Master

in the

GAA - Machine Learning Group
Department of Computer Engineering

September 2014

Declaration of Authorship

I, Carlos Asensio Pizarro, declare that this thesis titled, 'On the possibility to find coordinates by random features' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information,..."

Dave Barry

AUTONOMOUS UNIVERSITY OF MADRID

Abstract

Escuela Politécnica Superior
Department of Computer Engineering

Master

On the possibility to find coordinates by random features

by Carlos Asensio Pizarro

In this document we explore the possibility of using an image of the texture features on a surface for determining the absolute position of the device which extracted it, avoiding direct image comparison. As a result, we obtain a method which uses the relative positions of neighbouring features as a random code and determines the location of the device on the surface with a low probability of error and fast execution time.

Acknowledgements

I would like to express my gratitude to everyone who supported me during this a amazing project. Particularly to the professors for their advises and ability of illustrating concepts beyond the physical limits.

I would also like to thanks Ms. Patricia Ann Taylor-Hawksworth and Mr. Pedro Martínez Aguinaga for their inspiring comments on the document, and to Mario Daniel Ruiz for repairing the Digital Microscope.

Finally, I express my warm thanks to my project supervisor Mr. Kostadin Koroutchev for his support and guidance during the project.

Contents

List of Figures

List of Tables

Dedicated to my parents who have been a constant source of support and encouragement during the challenges of graduate school and life.

Chapter 1

Introduction to the thesis topic

1.1 Introduction

The relative positioning device, known as an optical mouse [14], is one of the most commonly used and widespread pointing devices. It is based on a comparison of the images of the texture of a surface in two consecutive moments. If the scanning frequency between frames is high compared to the speed of the mouse then a significant partial overlap among the consecutive frames exists, making it possible to compute the displacement between the frames and to show that movement on the screen. The displacement vector is not exact, but the adaptive human feedback compensates for the lack of precision of the estimation.

The problem arises when we consider an arbitrary surface and we want to know the absolute position of the point using a camera shot; i.e. when we want to determine the coordinates of a point given a frame of reference previously fixed on the surface (absolute position), instead of just determining the displacement vector from the previous position (relative position).

It seems quite obvious that by using the method of the optical mouse it is only possible to obtain the relative position of the optical device. Even if the starting point is well-known and we use the displacement vectors to iteratively find the position, the accumulated error will be large. Alternatively, we can use a printed surface, on which we have drawn special marks, and on which we can recover the position using these visible marks. These marks can be printed in a hardly visible way in order to preserve the integrity of the surface. However, in most cases it

is not possible, or not practical, to print special marks on the surface. So the question is: can we find a method, based on texture features, which is able to give us the absolute position of a point?

Nowadays, the most common techniques for obtaining the absolute position are based on the location of objects on the surface by identifying the features of the object in different images. Frequently, this is performed by combining algorithms such as RANSAC[3], Fast Approximate Nearest Neighbour Search (FLANN) [15] , K-Means [16], KD-trees [17], Bag of visual features (BOF) [18] and histograms where the feature descriptors are provided by algorithms like SIFT [19], SURF [20], ORB [21] and BRISK [22].

The problem with these methods is their tendency to find similarities rather than finding exact matches. This means that when applied on "random" surfaces with very similar texture features, the algorithms tend to make numerous errors and are usually unable to obtain accurate results. Some examples of textures where the performance and reliability of these algorithms is inaccurate are recycled paper, stucco ceiling and asphalt.

In the case of rigid objects, it is possible to perform the matching through the alignment of pairs of object features. Some techniques related to pair alignment of rigid objects are Pose Clustering [23] and Randomized Pose Clustering [24]. However, the performance of these algorithms in terms of error rate, memory and execution time are too high, limiting their applicability when handling textures instead of structures.

The methods based on direct sub-image comparison have even higher complexity than the pose clustering methods and therefore are not feasible for practical applications.

These limitations in the state of the art when working with "random" textures have forced the industry to develop alternative techniques. Most of these techniques are based on printing special patterns on the surfaces, with information about the position. For example, the company ANOTO has developed numerous patents to detect the position of an optical pen on a paper. Most of them are focused on devices and methods necessary to extract the features in order to obtain the absolute position code (US Patent 6,548,768 [25], US Patent 7,239,306 [26]), although this is only possible if the paper has been printed with a special ANOTO pattern (WO 2001/026032 [27], US Patent 6,854,821 [28]).

Some other companies such as HP, Sekendur and Sires have proposed alternatives to the ANOTO patterns (US 8100338 B2 [29], US 5852434 A [30] y US 7,991,191 B2 [4]). These alternative position codes, based on hardly visible marks, allow the identification of the absolute position of an optical device, but they have the same limitations as those seen in the ANOTO codes. A different approach, although with similar restrictions, is described in the patent WO 2006/137077 A1 [31]. In this patent the inventors, in order to find the absolute position of an optical device, use a surface divided into a plurality of non-overlapping fields which continuously cover the surface, and where each field is related to a unique colour.

The need to print predefined regular codes, uniquely coloured fields or cyclic polynomials to describe the absolute position on a surface has recently been questioned. The article written by Koroutchev and Koroutcheva in 2011 [5] shows that the relationship between the moments of different randomly distributed figures is enough to identify the absolute position with a high degree of reliability, increasing the freedom of designers who are not limited by having to draw pre-designed sequences of patterns when they want to allow the absolute positioning of devices over them.

This work shows that it is indeed possible to generalise the random codes described by Koroutchev and Koroutcheva [5] in order to find the absolute position of a device on a surface without drawing marks. Allowing the optical devices to find their coordinates using texture features for practically any surface, as long as it is not strictly periodic and has a uniform density of texture features covering it.

1.2 Objectives of the project

The aim of the project is to answer the question: Is it possible to find a method, based on texture features, which is able to provide the absolute position of a point?

In order to answer this question the following objectives are settled:

- 1: To find where the information is and how can it be coded.
- 2: To find out how to extract the best features for the code.
- 3: To identify effective structures to locate the coordinates.
- 4: To estimate the performance and reliability of the code.

- 5: To implement at least one method able to use the code for positioning
- 6: To calculate the performance and reliability of the code.

1.3 Structure of the document

The thesis is divided into five chapters and it is structured as follows:

- Chapter 1: Introduction to the thesis topic
- Chapter 2: State of the art
- Chapter 3: Theoretical estimation and developed methods.
- Chapter 4: Experimental results and analysis of results
- Chapter 5: Conclusions and future work

Chapter 2

State of the art

This Chapter offers a review of the previous work developed in the field of positioning; in particular, the positioning of devices over surfaces by optical means. The chapter is divided into five sections. The first provides a general vision of the problem of positioning (section 2.1), the second describes the methods for codifying position information on surfaces (section 2.2), the third analyses the methods for extracting the code features (section 2.3), the fourth illustrates the matching techniques (section 2.4), and finally a conclusion is provided (section 2.5).

2.1 The problem of positioning

Navigation and position determination have been great challenges since ancient times. However, natural human capabilities in combination with some discoveries such as the compass, brought to Europe by Marco Polo in the thirteenth century, and improvements in cartography since the fifteenth century have greatly simplified the problem.

Nowadays, the problem of human positioning around the globe is no longer a challenge, as a result of global technologies such as GPS [32] and radio-frequency signals [33]. However, devices and robots still have problems determining their indoor positioning.

In order to solve this problem, different solutions have been proposed. Most of them are based on relative positioning; for example, robots use self-motion information to estimate their position in relation to their initial position [34], but this

technology might lead to disorientation, as a result of the accumulated error or unregistered movements.

Alternatively, some devices use special guides and drawings on the floors and walls in combination with artificial vision systems to determine their absolute position, which are independent of the initial position of the robot. Some examples of this kind of device are line follower robots [35] and robots able to read QR or similar codes printed on the floor [36].

With the appearance of drones, some other advanced artificial vision technologies have been developed. Most of them are based on feature point detection and matching [37], by means of algorithms such as SIFT [19] and SURF [20], and image registration techniques [38].

2.2 Optical methods for encoding position over surfaces

A similar challenge, if not the same, is being faced by automatically positioning devices over surfaces; recording and digitalising this information with precision for future use.

As has happened with robot positioning, the most common techniques are based on relative positioning. By combining a well-known position as a reference, overlapping images of the surface, and measuring systems, a displacement vector is created (which is the technique used by the optical mouse for positioning [39]). This allows the estimation of position, but the accumulated error tends to be large and human feedback might be necessary.

2.2.1 Special codes

The difficulty of positioning an optical device over a surface, with precision and without external support, has forced the industry to draw special deterministic codes on the surfaces where they want to determine the position of the device. In particular, the ANOTO [28] and the CLUSPI [40] codes are widespread across the industry.

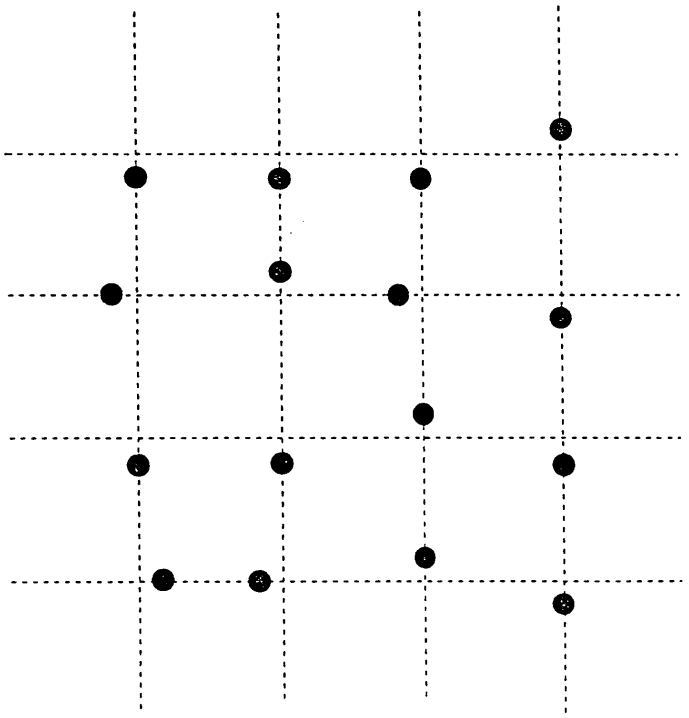

FIGURE 2.1: Example of the ANOTO pattern on an image extracted from [1]. The virtual and orthogonal raster lines, where the dots are drawn, are denoted by dashed lines.

2.2.1.1 ANOTO codes

The ANOTO group have developed more than 300 patents to protect this technology. It is based on printing a pattern of hardly visible micro-dots on a surface, which are recognised by the ANOTO pen [41], [42].

The ANOTO pattern was first described in a US patent in 2003 [1]. The pattern defines some orthogonal virtual raster lines, and identical marks with a certain displacement among the raster lines from its nominal position (which correspond to the intersection between each pair of raster lines); the displacement being a number in the range (1/8,1/4) of the distance between two parallel raster lines (usually 0.3mm).

As each mark has 2^2 possible positions on the grid, it contains 2 bits of information. Therefore, by scanning a grid with 16 (4x4) different marks, as shown in the example in figure 2.1, a code with 32 bits is obtained, which is enough to uniquely codify 2^{32} different positions.

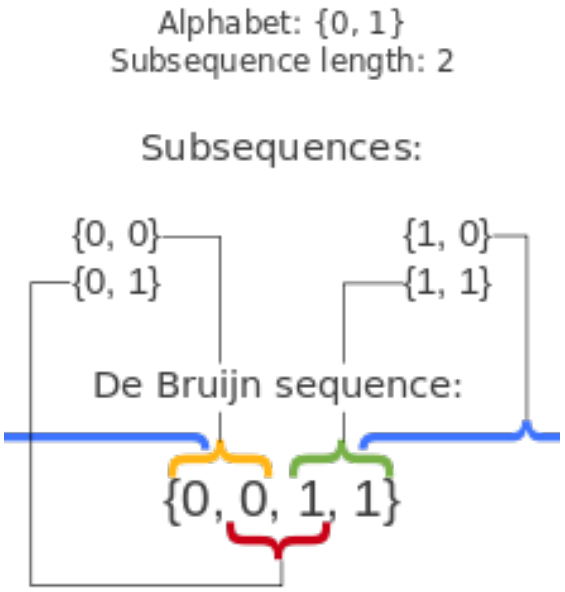

FIGURE 2.2: An example of a De Bruijn sequence using a binary alphabet and sub-sequences of size 2 extracted from [2].

Nevertheless, in order to avoid possible gaps in some regions of the paper produced by non-overlapping regions, the ANOTO pattern employs a specific De Bruijn sequence [43]. The De Bruijn sequence is the shortest cyclic sequence that includes all the sub-sequences of a fixed length of an alphabet (an example using a binary alphabet and sub-sequences size two is shown in figure 2.2).

In particular, the sequence used by the ANOTO pattern uses one bit of each cell to codify the x position and the other bit to codify the y position. The x coordinate has a value related to the main sequence, which, for the recommended 6x6 code length is $\{\ 0,0,0,0,0,0,\ 1,0,0,1,1,1,\ 1,1,\ 0,1,\ 0,0,\ 1,0,\ 0,0,\ 0,1,\ 1,1,\ 0,1,\ 1,1,\ 0,0,\ 1,0,\ 1,0,\ 1,0,\ 0,0,\ 1,0,\ 1,\ 1,\ 0,1,\ 1,0,\ 0,1,\ 1,0,\ 1,0,\ 1,1,\ 1,1,\ 0,\ 0,0,\ 1,1\ \}$ [3]; it is cyclically repeated. Each x+1 column uses the same main sequence, but the sequence is displaced, as shown in figure 2.3. It is possible to use the relationship between the six offsets to uniquely code 54^6 positions (by restrictions of the patent the offset belong to the interval [5,58]). The code operates the same way on the y axis.

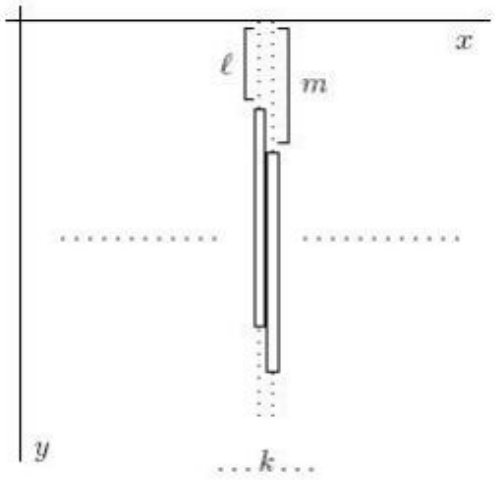

FIGURE 2.3: An example of the offset between x columns main sequences extracted from [3].

2.2.1.2 CLUSPI codes

Like the ANOTO pattern, which codifies each zone by means of cyclic polynomials, the CLUSPI codes use arrangements of overlapping virtual blocks associated with position information.

Each virtual block of the CLUSPI code is composed of a group of cells, where each cell contains a code of 2 bits (a bit codifying the X value and a bit codifying the Y value); this is done by means of one of the four possible alphabets described in the CLUSPI patent [4], which are shown in figure 2.4.

The arrangement of the cells in the virtual code are described by the condition of satisfying the equations (2.1), (2.2) and (2.3).

$$v(i,j) = b_m \tag{2.1}$$

$$v(i+1,j) = b_{m+1} \tag{2.2}$$

$$v(i+1,j) = b_{m+a} \tag{2.3}$$

$v(i,j)$ being the binary value at the position i, j on the grid, b an array with a predefined bit sequence of size n, b_m each one of the $m = (m = 0tom = n-1$ possible values of the sequence, and a a predefined offset bigger or equal than 2.

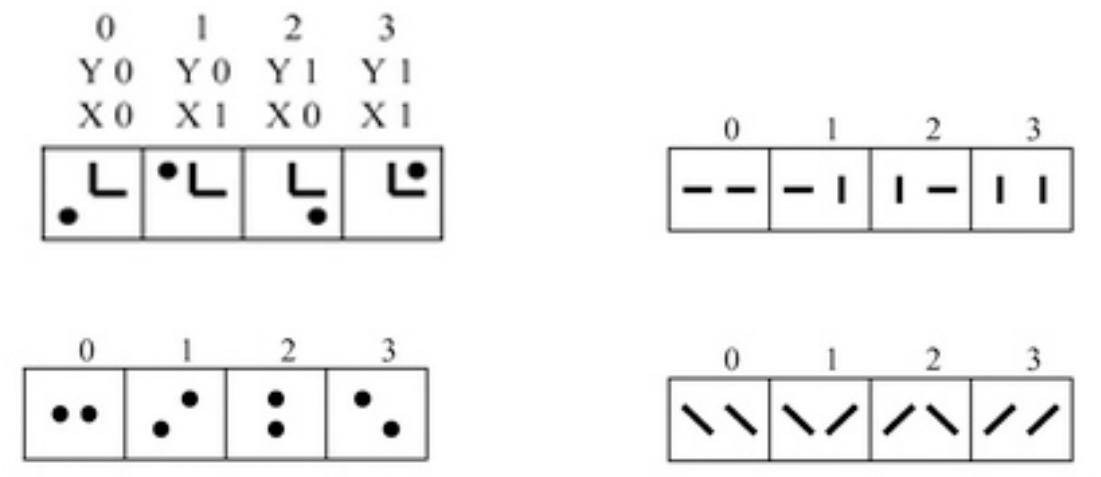

FIGURE 2.4: Some examples of the CLUSPI symbols used for coding extracted from [4].

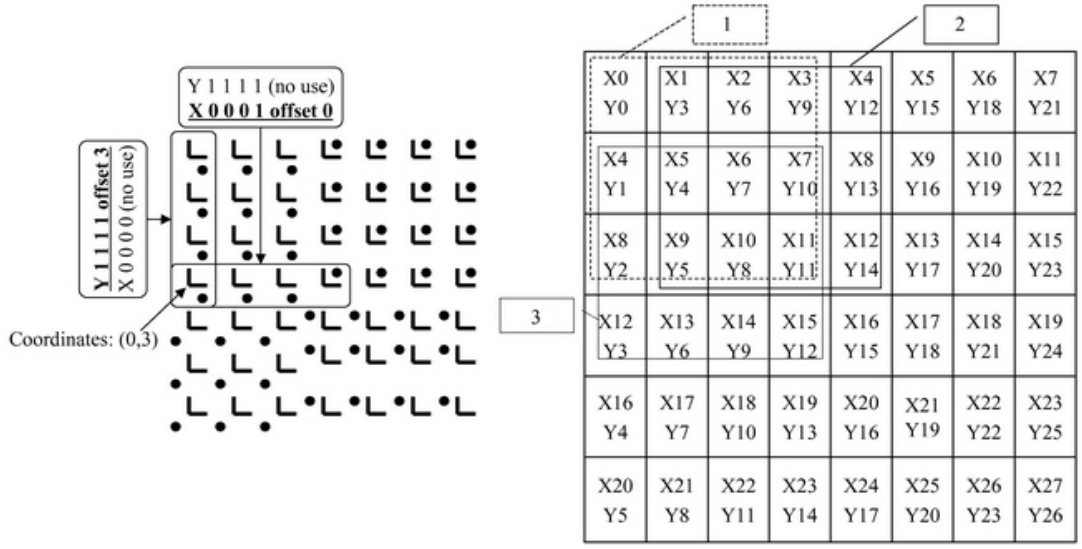

FIGURE 2.5: Example of the CLUSPI code extracted from [4].

2.2.2 Random Coding

In the previous sections, we have seen that it is possible to draw special deterministic patterns to encode position information. However, it is not necessary to rely on deterministic codes in order to choose and draw the patterns. Instead, we can use the properties of random numbers to define the codes. This is called random coding.

2.2.2.1 Basis of Random coding

Let's suppose that we have an infinite white surface with randomly chosen numbers written on it, and we draw a square on the surface. As the random numbers have the property of being equiprobable and independent of each other, then the probability P_A, of drawing on the surface another square with the same area and containing the same group of numbers seen in the area of the first square, can be described by the equation (2.4)

$$P_A = 1/k^n \tag{2.4}$$

where n represents the number of random numbers in the area and k represents the number of discrete possible values that can be chosen for each random number.

Thus, in the example displayed in figure 2.6, the probability of finding the same sequence observed inside the green rectangle, just by chance, would be $1/10^{15}$, since $k = 10$ and $n = 15$. Assuming that an A-4 paper contains 10^4 numbers, then the code has a high enough probability to uniquely codify its position, in terms of page number and position within the page, on more than 10^{10} pages.

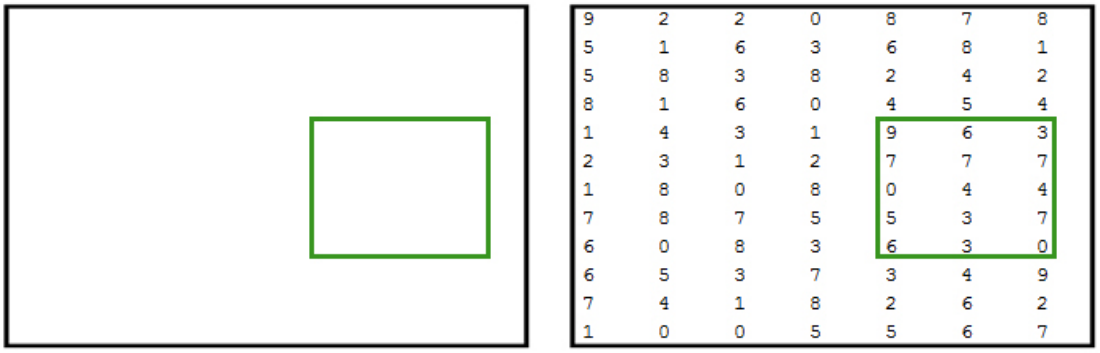

FIGURE 2.6: Example of random code in an image.

2.2.2.2 Random Coding for surface design

Even though the random codes work quite well for describing position information, the idea of filling the surfaces with random numbers does not seem very appealing to designers. Therefore, a new and less invasive way of random coding had to be developed.

In 2011 Koroutchev and Koroutcheva presented a model of random coding based on figure orientation [5]. This method of coding uses the relationship between the rotations of n different figures to generate the random codes necessary to describe each zone on the surface. Some examples of the rotations obtained from the figures are shown in figure 2.8. The rotation of the different figures is determined in relation to a reference system fixed and oriented in function to the central figure. An example of the relationship between the rotation of the figures is shown in figure 2.7.

This code gives the possibility of uniquely codifying every position on a din A4 sheet with just 8 scanned figures,and the probability of confusing the codes just by chance is smaller than 10^{-5}.

In terms of performance of the code, the memory cost of storing grows linearly with the number of figures (i.e $O(n)$) and the computational time for finding a stored code is constant(i.e $O(1)$), if a HashTable is used to save them.

FIGURE 2.7: Example of random coding based on image orientation extracted from the article [5].

2.3 Feature detection, extraction and description

Once we have reviewed the different approaches seen in literature to codify the absolute position on a surface, it is important to analyse the methods which allow the detection, the extraction, and the description of the features on the surface. The selection of these methods is critical for the detection of the code because a poor choice of the features may lead to unreliable codes, erroneous matches, and additional memory and execution times.

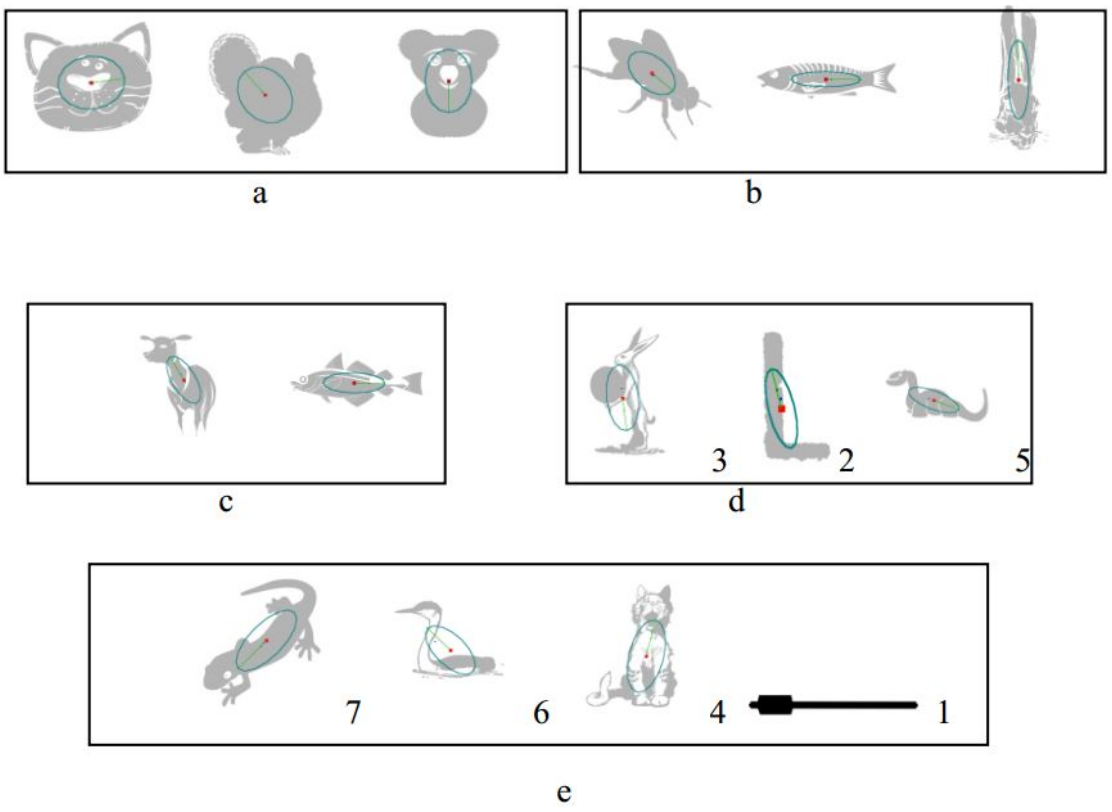

FIGURE 2.8: Examples of figures and their orientations.

2.3.1 Feature detection

In order to detect the features of an image, there are two main approaches: the first is based on edge detection, and the second is focused on blob detection. The methods aimed at edge detection use the discontinuities and the sharp changes in the brightening of the image regions to identify the areas of interest. On the other hand, the blob detection methods are targeted at identifying the regions of the image with distinctive properties, when compared to the surrounding areas.

2.3.1.1 Edge detection methods

Prewitt filter The Prewitt filter developed in 1970 [44] is one of the first filters developed for image processing. These two filters are focused on enhancing the edges of an image, to simplify the recognition of objects.

These filters are described by two convolution kernels of size $3x3$, one focused on highlighting vertical edges (2.5) and the other focused on the horizontal (2.6).

$$S_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \tag{2.5}$$

$$S_y = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \tag{2.6}$$

By applying both kernels to the image, it is possible to calculate the response G_r (2.7) of each point of the image $f(x, y)$ and the angle of the response ϕ_r (2.8). However, the filter is too reactive to all the possible edges, producing overreactions when it is faced with high frequency variation in the image.

$$G_r = \sqrt{(S_x * f(x,y))^2 + (S_y * f(x,y))^2} \tag{2.7}$$

$$\phi_r = atan2(S_x * f(x,y), S_y * f(x,y)) \tag{2.8}$$

Sobel filter The Sobel filter was presented by Sobel in 1968 at Stanford Artificial Intelligence Laboratory, but it was never officially published. These two filters are similar to the ones described by Prewitt, and also have a lot of problems when dealing with noise.

These filters are described by two convolution kernels of size $3x3$, one focused on highlighting the vertical edges (2.9) and the other focused on the horizontal edges (2.10).

$$S_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{2.9}$$

$$S_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \tag{2.10}$$

As with the Prewitt filters it is possible to calculate the response, and the angle of the edges, by applying the equations (2.7) and (2.8).

Gabor filters The Gabor filters are a family of filters that use a response to frequency and orientation similar to that seen in the visual cortex of mammals [45] [46]. These filters have been successfully used for edge detection and for texture processing [47] and segmentation [48]. Some examples of Gabor filters used for texture classification are shown in figure 2.9.

The convolution Kernel of these filters $gb(x, y)$ (2.15) is obtained by multiplying a complex sinusoidal planewave function $s(x, y)$ (2.11), with certain frequencies (u_0, v_0) and phase φ, known as the carrier, with a Gaussian bi-dimensional function $w_\theta(x, y)$ (2.14), with a certain orientation θ and scale parameters (K,a,b), known as the envelope.

$$s(x, y) = e^{i(2\pi(u_0 x + v_0 y) + \varphi)} \tag{2.11}$$

$$(x - x_0)_\phi = (x - x_0)cos\theta + (y - y_0)sin\theta \tag{2.12}$$

$$(y - y_0)_\phi = -(y - y_0)sin\theta + (y - y_0)cos\theta \tag{2.13}$$

$$w_\phi(x, y) = Ke^{(-\pi(a^2(x-x_0)_\theta^2 + b^2(y-y_0)_\theta^2))} \tag{2.14}$$

$$gb(x, y) = s(x, y) * w_\theta(x, y) \tag{2.15}$$

Canny edge detector The canny edge detector [7] is a multi-stage edge detection algorithm. The algorithm combines four stages (noise reduction, gradient enhancing, non-maximum suppression and hysteresis thresholding), in order to get a binary image with only the edges of the image.

The first stage, noise reduction, is performed by applying a Gaussian filter (2.21) to the input image $f(x, y)$. Then, the second stage, gradient enhancing, highlights the edges of the image by convolving an edge detection operator, such as the Prewitt 2.3.1.1 or the Sobel operators 2.3.1.1. The third step, non-maximum suppression, is focused on removing spurious points from the highlighted image by following the maximum gradient of the edges. Finally, the last stage, hysteresis thresholding, employs an adaptive threshold to obtain the binary image with only the detected edges. Figure 2.10 shows the result of the canny edge detector.

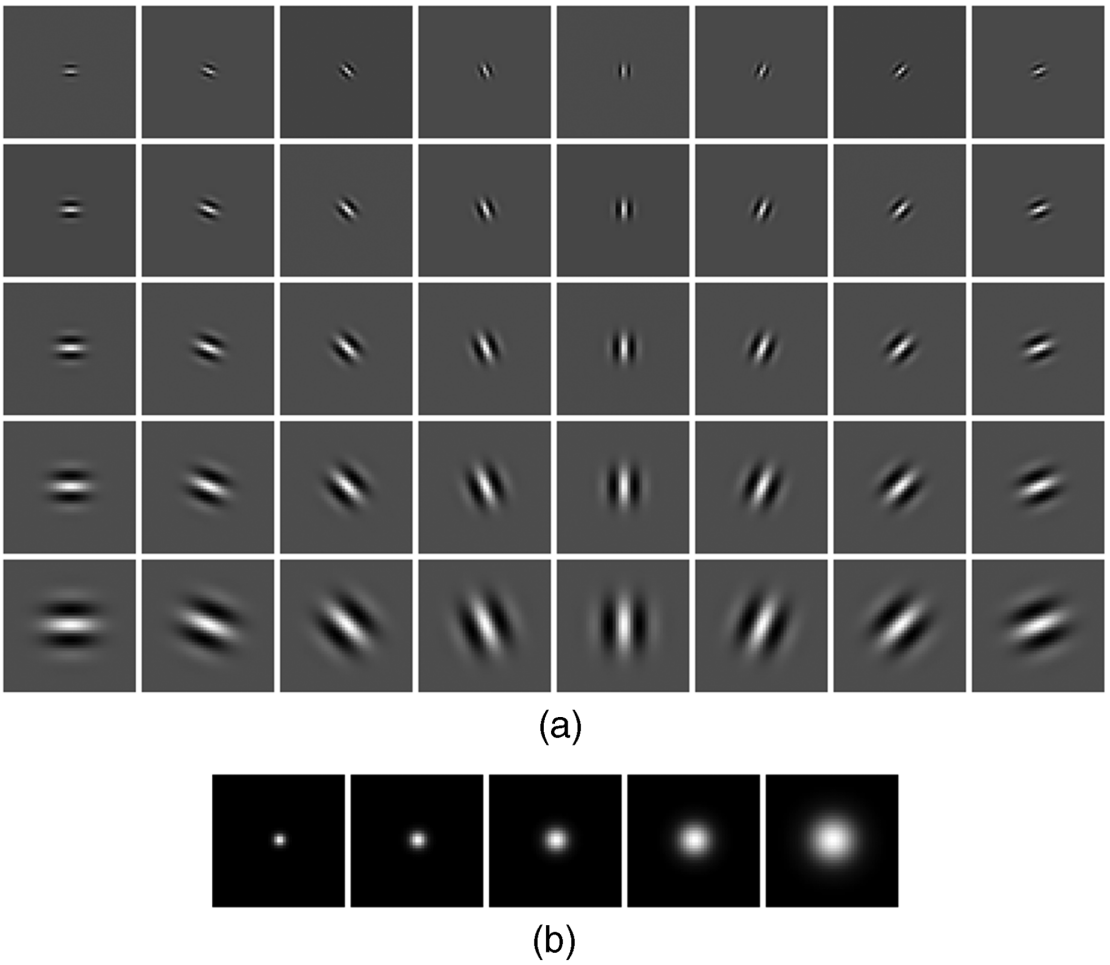

FIGURE 2.9: Example of Gabor filters used for texture classification displayed in the article [6]. The image (a) presents the bank of oriented Gabor filters at different scales and the image (b) shows the Gaussian Kernels at each of the scales used to create the Gabor filters.

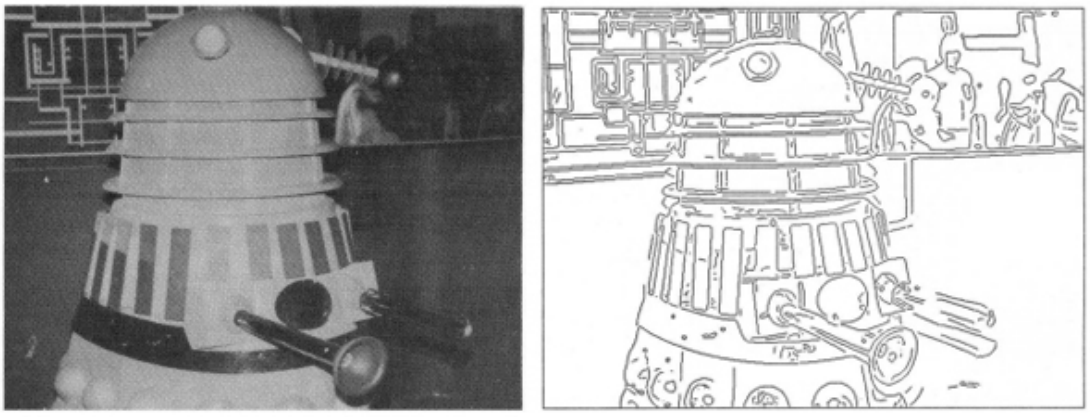

FIGURE 2.10: Example of application of the canny edge detector, extracted from the article [7]. The left image is the input image and the right image is the resulting image after executing the algorithm.

2.3.1.2 Blob detection methods

Laplacian of Gaussian (LoG) The Laplacian of Gaussian is one of the oldest and most frequently used band pass filters for blob detection, and it is optimal for detecting symmetric blobs at a fixed scale.

It is obtained by applying the Laplacian operator, described by equation (2.17), to an image previously smoothed with a Gaussian convolution filter (given by the equation (2.21)) at a certain scale σ. However, the convolution kernel of this filter, whose 3D shape (shown in figure 2.11) is similar to an inverted Mexican hat, can be pre-calculated by means of the equation (2.19)

$$L(x, y; \sigma) = g(x, y, \sigma) * f(x, y) \tag{2.16}$$

$$\nabla^2 L = L_{xx} + L_{yy} \tag{2.17}$$

$$h_g(n_1, n_2, \sigma) = e^{\frac{-(n_1^2 n_2^2)}{2\sigma^2}} \tag{2.18}$$

$$LoG(n_1, n_2; \sigma) = \frac{n_1^2 + n_2^2 - 2\sigma^2 h_g(n1, n2, \sigma)}{2\pi\sigma^6 \sum_{n_1} \sum_{n_2} h_g} \tag{2.19}$$

where σ represents the scale of the filter and n_1, n_2 represents the size of the components x and y of the convolution kernel.

Since the response to the LoG filter is highly sensitive to scale (indeed, it has the strongest response to circular blobs of size $\sqrt{2\sigma}$), if the blob size is unknown or highly variant it is necessary to calculate scale invariant blobs. As described by [49] it is possible to calculate scale invariant blobs $\hat{x}, \hat{y}$ by applying the normalised equation (2.19) at different scales $\hat{\sigma}$ and calculating the three dimensional maxima of the points. Formally, it is described by the equation (2.20).

$$(\hat{x}, \hat{y}; \hat{\sigma}) = argmaxminlocal_{(x,y;\sigma)}(\nabla^2_{norm} L(x, y; \sigma)); \tag{2.20}$$

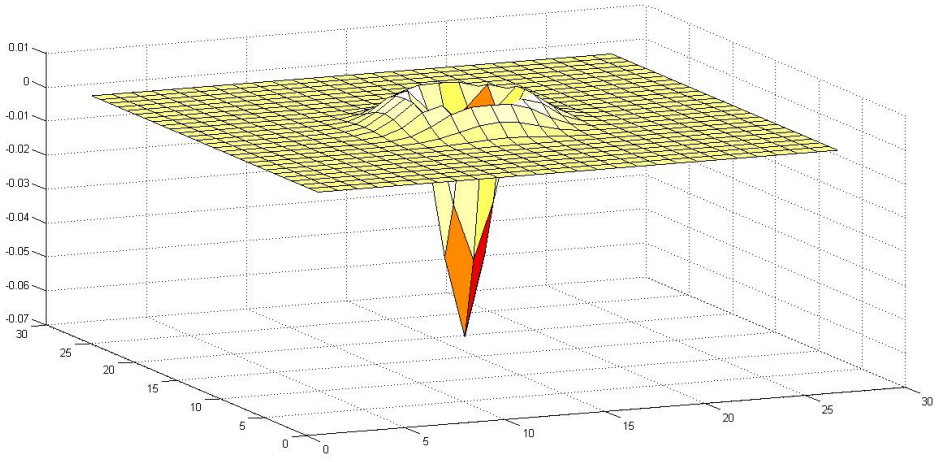

FIGURE 2.11: Example of the 3D shape of a LoG convolution kernel with $n_1 = 27$, $n_2 = 27$ and $\sigma = 1.5$.

Difference of Gaussians (DoG) Instead of using the LoG filter, it is possible to apply the Difference of Gaussian (DoG) to obtain the blob features of an image $f(x, y)$.

This convolution filter, which imitates the mechanism used by retinal neurons to extract visual features [50], can take a similar shape to that observed in the LoG kernel by properly adjusting the scales σ_1 and σ_2 of the Gaussians. However, it is easier to compute. The DoG filter is given by the equation

$$g(x, y, \sigma) = \frac{1}{2\pi\sigma^2} * e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.21}$$

$$DoG(n_1, n_2, \sigma_1, \sigma_2) = g(n_1, n_2, \sigma_1) * f(x, y) - g(n_1, n_2, \sigma_2) * f(x, y) \tag{2.22}$$

2.3.2 Feature extraction

After enhancing the image to highlight the blob-like features, it is necessary to identify and label the most prominent features of the image. This process is performed in two steps: thresholding and applying a labelling algorithm.

Thresholding The threshold is one of the basic operators in signal and image processing. The thresholding process segments the image into two classes, generating a binary image. The basic thresholding process, where the pixels are classified according to a constant Theta, can be modeled by the equation (2.23).

$$th(x, y, \Theta) = f(x, y) > \Theta \tag{2.23}$$

Nevertheless, when an image is being thresholded the lighting conditions might not be uniform and the basic thresholding method may produce an erroneous segmentation of the image.

As has been described in [8], when the threshold method is facing a non uniformly lightened image, it is more accurate to apply an adaptive thresholding method instead (as shown in figure 2.12). The adaptive thresholding method works similarly to the basic thresholding, but selects the threshold value Θ for each pixel of the image as a function $lvl(f(x, y))$ of its neighbours. The method is given by the equation (2.24).

$$ath(x, y) = f(x, y) > lvl(f(x, y)) \tag{2.24}$$

Labelling algorithm Once the binarised image is obtained, the different connected components, which represent each surface feature, must be labelled. This can be performed by means of different algorithms and techniques; some of these, such as a recursive labelling algorithm and a row-by-row labelling algorithm, are described in [51] .

However, one of the most commonly used methods is the border following algorithm based on topological structural analysis proposed by [9], since it simultaneously provides hierarchical and topological information about the blobs (figure 2.13).

2.3.3 Invariant feature points detectors and descriptors

Instead of applying different methods to obtain the features, some well-known algorithms for invariant feature detection and description can be applied. Probably,

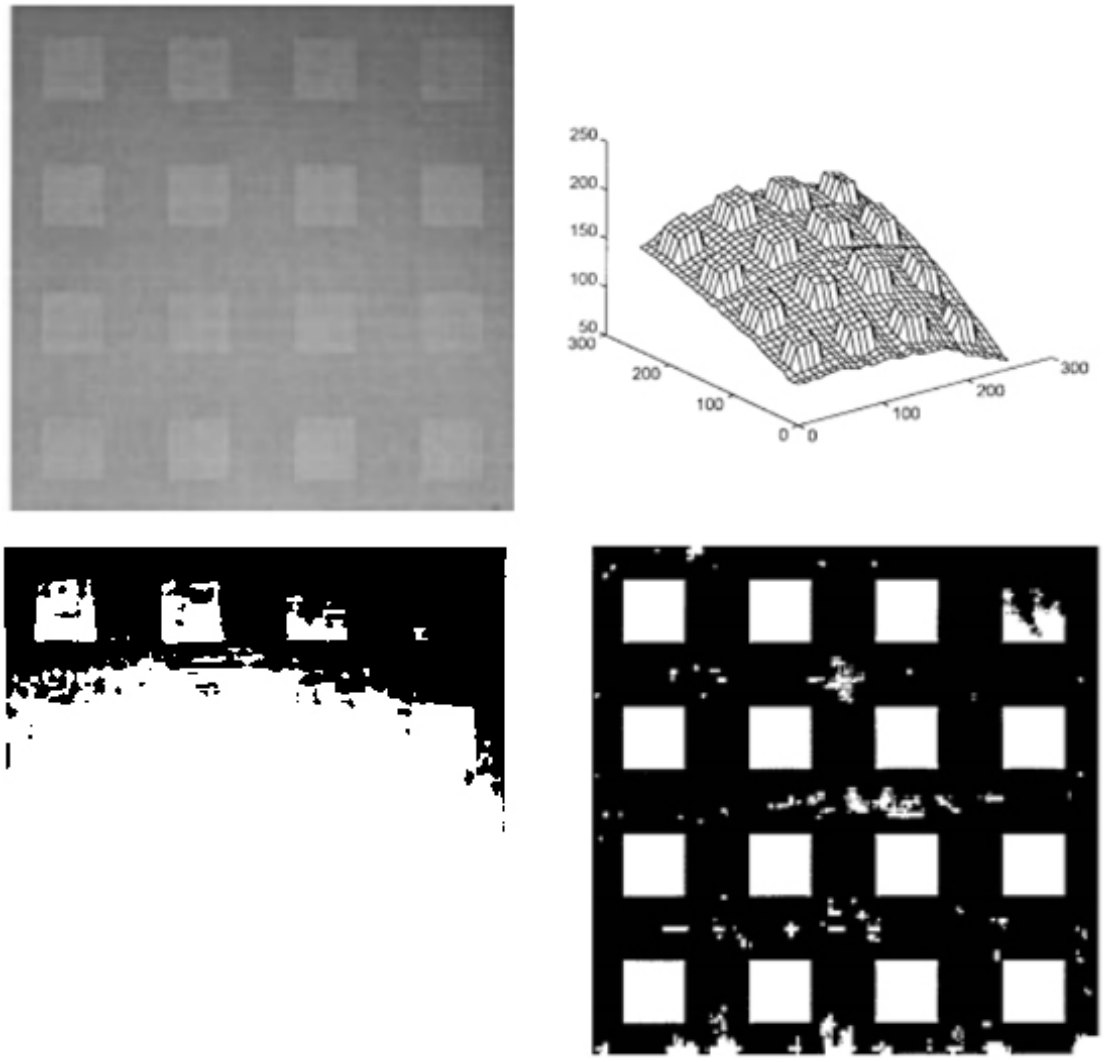

FIGURE 2.12: Examples extracted from [8] of static thresholding and adaptive thresholding. The top left image shows the original image, the top right figure represents the 3D model of the image, the bottom left image displays the image thresholded with a fixed value and the bottom right picture shows the image thresholded with an adaptive threshold.

the most widely used are the Scale Invariant Feature Transform (SIFT) [19] and the Speeded Up Robust Features (SURF) [11], [20]. However, there are some others such as Binary Robust Invariant Scalable Keypoints (BRISK) [22], Fast Retina Keypoins (FREAK) [52], Features from Accelerated Segment Test (FAST) [53], and Oriented FAST and rotated BRIEF (ORB) [21].

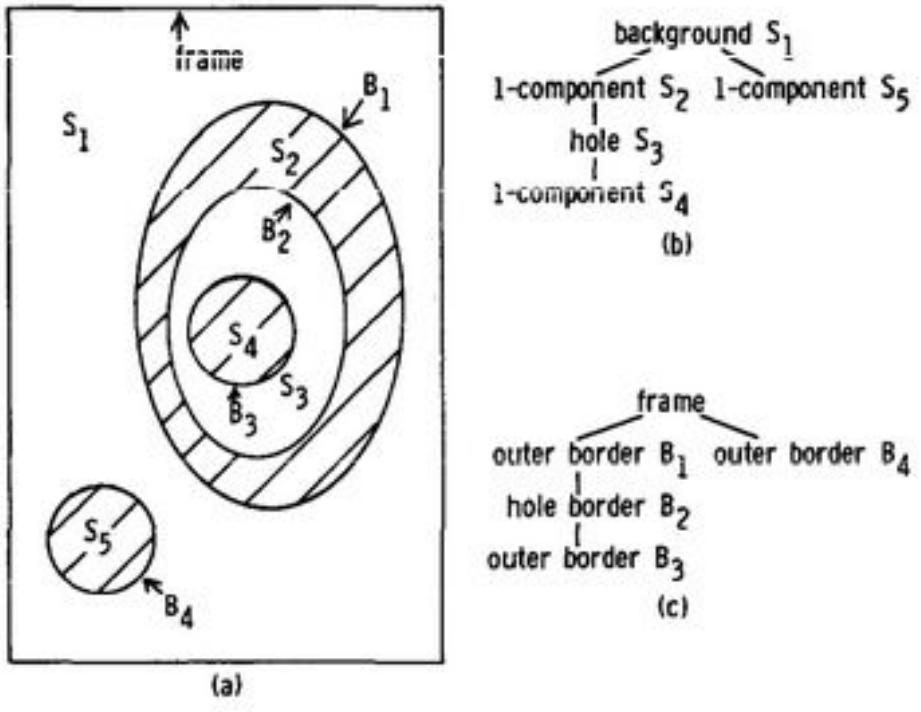

FIGURE 2.13: Example of execution of the border following algorithm proposed by [9] .

2.3.3.1 SIFT

The Scale Invariant Feature Transform [19], also known as SIFT, is one of the oldest and most effective algorithms for invariant feature points detection and description; although it is one of the slowest, as was demonstrated in [54].

The algorithm is divided into four main steps: Scale-space extrema detection, Keypoint localization, Orientation assignment and Keypoint description.

The first step, Scale-space extrema detection, is focused on searching over the scales of the image in order to identify the candidate feature points. This is performed by applying some DoG kernels to the image, where the σ values of the filters are chosen as adjacent values of the Gaussian Pyramid [55] (the pyramid is exemplified in figure 2.14). Then, the keypoints are extracted by choosing the local maxima and minima of the points in the three dimensional feature model obtained, which is formed by the 26 neighbours of the pixel in the nearest σ scales of the pyramid, as is shown in figure 2.15.

Following the Scale-space extrema detection, the second step, Keypoint localization, is responsible for reducing the number of unstable points. It is performed

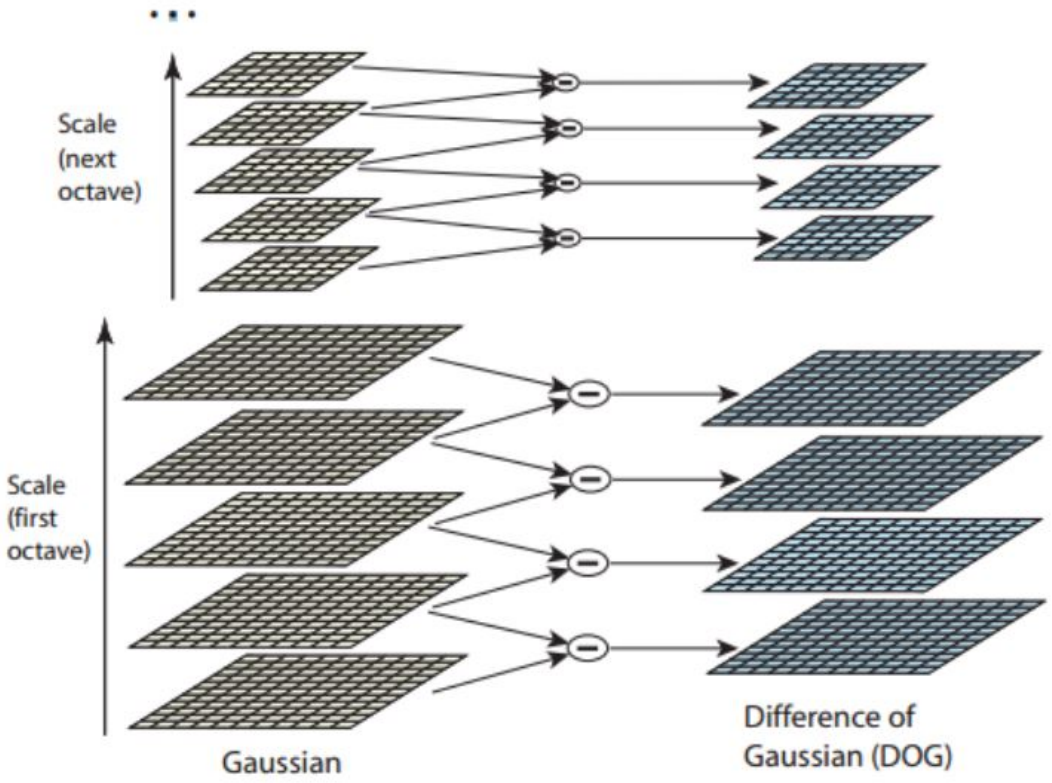

FIGURE 2.14: Image extracted from [10], which describes the process to apply the DoG through the Gaussian pyramid. For each σ scale the images are smoothed with the corresponding Gaussian; then, the two smoothed images are subtracted to produce the DoG image. Once the DoG image is extracted, the σ scale is duplicated to obtain the next octave of the pyramid and the process is repeated.

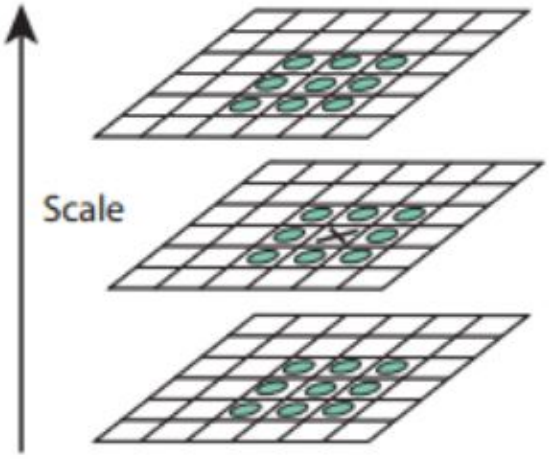

FIGURE 2.15: Example extracted from [10] of the 26 neighbours of the pixel compared to the nearest DoG scales in order to obtain the local maxima/minima.

by interpolating the near data, discarding low contrast points, and removing edge responses.

In the approach described in 1999 [19] no interpolation was performed and only the biggest responses were chosen. However, in 2004, Lowe [10] proposed the use of an interpolation based on the quadratic Taylor expansion of the Difference-of-Gaussian scale-space function g (2.21), given by equation (2.25), in order to improve the accuracy of the location of the features.

$$D(x) = D + \frac{\partial D^T}{\partial x^2}x + \frac{1}{2}x^T\frac{\partial^2 D}{\partial x^2}x \tag{2.25}$$

where $D(x)$ and its derivatives are evaluated at the candidate keypoint and $x = (x, y, \sigma)$. The extremum $\hat{x}$, is obtained by calculating the derivative of (2.25) at x and setting it to 0, obtaining the equation (2.26).

$$\hat{x} = \frac{\partial^2 D^{-1}}{\partial x^2}\frac{\partial D}{\partial x} \tag{2.26}$$

Then, the low contrast keypoints are eliminated because the probability of being unstable is high.

Finally, the keypoints which belong to the edges are removed; this is an important step since DoG is very sensitive to the edges. The process of edge elimination is performed using the Hessian matrix H and a curvature threshold ratio r, where the point is removed if the equation (2.27) is not satisfied.

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r} \tag{2.27}$$

After obtaining the stable Keypoints, the points must be oriented in order to obtain rotation invariance. This is performed by analysing the gradient around the point by means of the equations (2.28) and (2.29)

$$m(x, y) = \sqrt{(f(x+1, y) - f(x-1, y))^2 + (f(x, y+1) - f(x, y-1))^2} \tag{2.28}$$

$$\theta(x, y) = tan^{-1}\frac{f(x, y+1) - f(x, y-1)}{f(x+1, y) - f(x-1, y)} \tag{2.29}$$

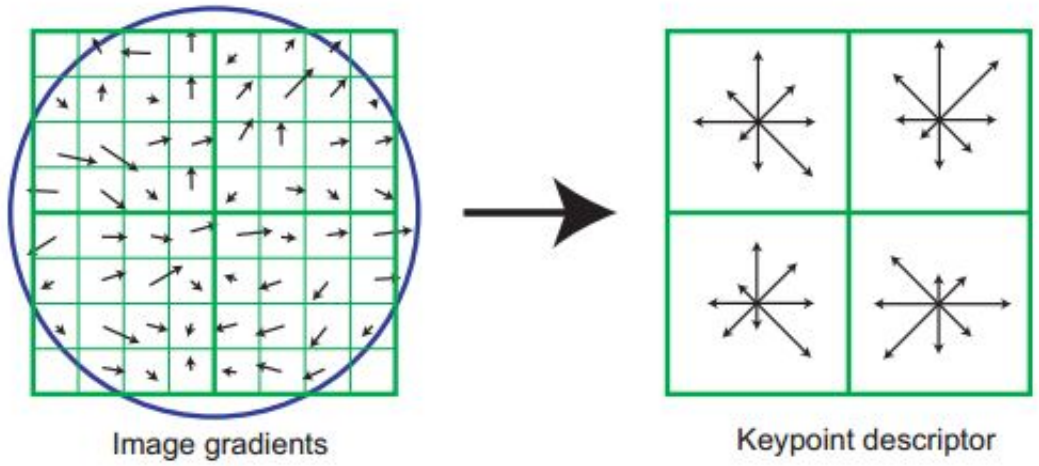

FIGURE 2.16: Image extracted from [10], where an 2x2 descriptor set is obtained from a 8x8 set of samples.

where $f(x, y)$ represents each image sample at this scale, $m(x, y)$ the gradient magnitude, and $\theta(x, y)$ the orientation.

Finally, the last step, the keypoint description, is accomplished by calculating the gradient of the surrounding areas of the keypoint, weighted with a Gaussian window. An example of the descriptor is shown in figure 2.16.

2.3.3.2 SURF

In 2006 Bay *et al.* presented an alternative method to SIFT. Their method, called Speeded Up Robust Features (SURF) [11], outperforms SIFT in both execution time and resilience to affine transforms.

This optimization of the SIFT algorithm is based on the Hessian of the Gaussian matrix H over integral images [56] for keypoint location. The Hessian of the Gaussian matrix is given by the equation (2.30)

$$H(x, y; \sigma) = \begin{bmatrix} L_{xx}(x, y; \sigma) & L_{xy}(x, y; \sigma) \\ L_{xy}(x, y; \sigma) & L_{yy}(x, y; \sigma) \end{bmatrix} \tag{2.30}$$

where x, y represents the position of the point, σ represents the scale, and $L_{xx}(x, y; \sigma)$ is the convolution kernel of the Gaussian second order derivative $\frac{\partial^2 g(\sigma)}{\partial x^2}$.

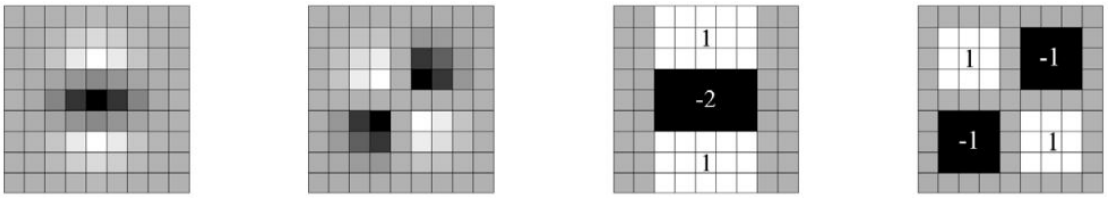

FIGURE 2.17: Example extracted from [11] of the Gaussian second order derivatives, and their approximation using box filters.

SURF improves the performance in both execution time and accuracy. Also, the use of the determinant of the Hessian avoids the use of a different measure for the location and scale, as was observed using the Hessian-Laplace operator [57].

For further improvement in performance speed, the SURF algorithm substitutes the Hessian of the Gaussian convolution kernels with box filters, named Fast-Hessian kernels (some examples of these filters are shown in figure 2.17), which reduce the computational cost. In combination with the integral images the use of Fast-Hessian kernels allows the application of the filter at different σ scales without losing performance. By using this approximation, the calculation of the determinant of the Hessian matrix $det(H_{approx})$ is simplified to the equation (2.31)

$$det(H_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2 \tag{2.31}$$

where w is a constant dependent of the σ scale, and D_{xx} is the Fast-Hessian kernel associated to the L_{xx}.

Finally, in order to identify the keypoints, the algorithm performs a non-maxima-suppression in a 3x3x3 neighbouring area.

After identifying the keypoints, the SURF algorithm creates a descriptor for each one of them. This is implemented in two steps.

The first step is the Orientation Assignment. To implement this step, the SURF algorithm uses the highest averaged response of the Haar-wavelet in the x and y directions around the point, at different scales.

The second and last step of the descriptor, the extraction of the descriptor, is executed by segmenting the neighbouring regions of each keypoint, and describing each of them with a four dimensional vector $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|$, where v is the vector, and d_x represents the response of the region to the Haar-wavelet in the x direction.

2.4 Feature and template matching

After extracting the features, and if they are not arranged using a specific pattern, such us those described by ANOTO 2.2.1.1 or CLUSPI 2.2.1.2, it is necessary to identify and to locate them.

With the aim of solving this problem, some methods have been developed; most of which have focused on object recognition. These methods are organised into two main approaches, known as Feature Based Matching and Template Based Matching.

2.4.1 Feature Based Matching

The Feature Based Matching method is a bottom up approach. It is focused on finding the best match for most, or all of the described feature points. The methods based on this approach have the advantage of being resilient to affine transformations and partial occlusion.

2.4.1.1 Brute Force

The Brute Force algorithm is the simplest and most effective, but it is the slowest one. Because of the low computational time performance ($O(n * m)$, where n is the number of image keypoints and m is the number of template points) it is not used unless the number of matching and template points is relatively small. It is based on comparing (using a measuring distance such as the Euclidean or Manhattan distances) each possible keypoint descriptor of the image with each feature descriptor of the templates.

2.4.1.2 RANSAC

With the aim of improving the computational performance of the Brute Force Matcher, the RANdom SAmple Consensus(RANSAC) was developed by Fischler in 1981 [58]. The RANSAC algorithm used to outperform the Brute Force Matcher and provide a better fitting to the template.

This probabilistic algorithm is based on the hypothesis that the image contains inliers, points which can be fitted to the model, and outliers, which cannot be fitted; and the probability of finding inliers is comparatively high when compared with the probability of finding outliers.

Since the probability of finding inliers is high, it iteratively selects random groups of descriptors from the image and fits them to the template. Then, when the number of correspondences between the image and the template is over a threshold, the algorithm finishes.

2.4.2 Template-based Matching

Conversely, the Template-based Matching method follows a top-down approach. It is based on finding the spatial or geometrical relationships between multiple features, ignoring the descriptor of each individual keypoint. This approach is generally faster than the feature-based method.

2.4.2.1 Pose Clustering

The pose clustering algorithm is based on a comparison of the relationships between the poses of the keypoints in the image, with the pose of the features in the model [23]. This algorithm has been successfully applied to object detection [24] and aerial photography alignment [12].

The algorithm as described in [24], performs the following steps: first, two random image points (v_1, v_2) are extracted; next, for all pairs of features (u_1, u_2) of the model, the matching points (v_3, u_3) are found, and the pose which align the group $\gamma = \{(v_1, u_1),\ (v_2, u_2), (v_3, u_3)\}$ is determined. Finally, the clusters of aligned groups are obtained. The process is repeated several times.

An example of the execution of the method for aligning an aerial image with a map is shown in the figure 2.18.

2.4.2.2 Geometric Hashing

Geometric Hashing, first described by Schwartz in [59], is an algorithm that organises the information of the model in buckets of a hash table. It chooses as key

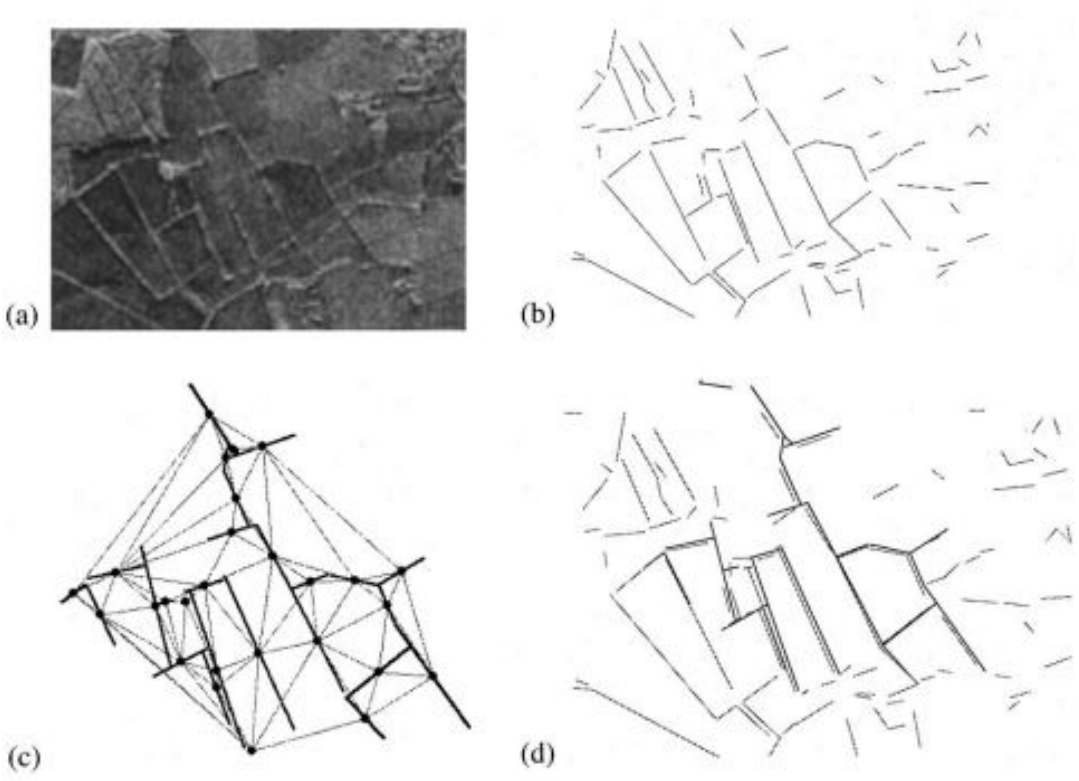

FIGURE 2.18: Image extracted from [12], where an aerial image (a) is fitted to a model (b); the image (c) shows the extracted edge features from a and (d) shows the result of aligning b and d using pose clustering.

the base formed by two points of the model and indexes the data of the coordinates of the surrounding points. Then in order to recognise an image the hash table is accessed and the index of the corresponding buckets are checked. If there are enough correspondences the results are verified and the algorithm finishes. A diagram describing the stages followed by the algorithm is shown in figure 2.19.

2.5 Conclusion

In this chapter we have reviewed three different coding methods (Anoto codes, Cluspy codes and random coding) seen in the literature to draw predefined patterns over surfaces in order to allow the positioning of devices on them.

In addition, different approaches for feature detection, extraction and description have been described. These approaches are based on edge detection techniques, blob detection techniques and invariant feature point detectors.

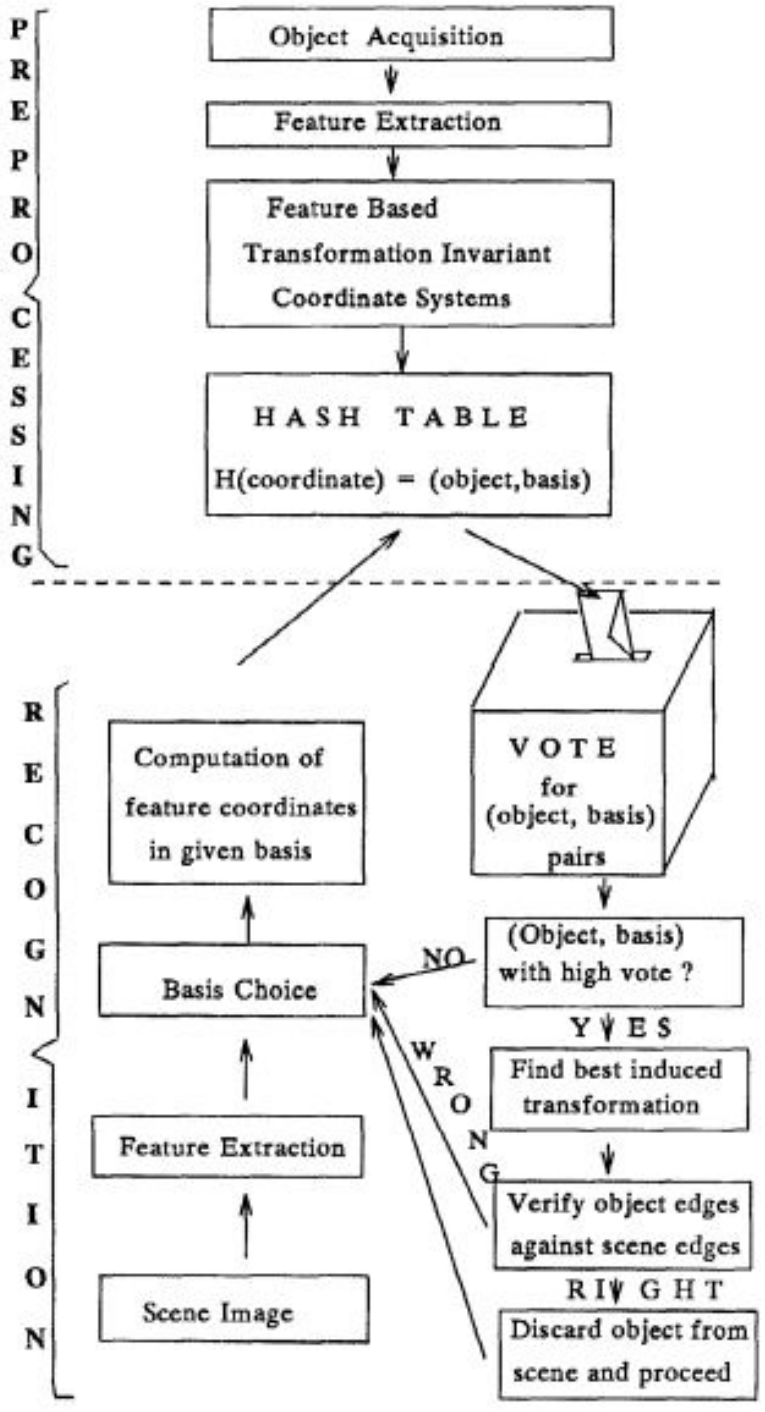

FIGURE 2.19: Flowchart of the Geometric Hashing Algorithm extracted from [13].

Finally, some methods for matching are described. These method are organised in two main matching approaches, feature matching that is slower but resilient to noise, occlusion and affine transforms, and template matching that is faster but more susceptible to noise.

Chapter 3

Outline of the method

This Chapter provides an outline of the proposed method and the theoretical estimations of the behaviour of the code. The chapter is divided into three sections. The first provides a general vision of the different approaches tried to obtain the proposed method and gives a theoretical estimation of its behaviour (section 3.1). The second outlines the different phases of the proposed method (section 3.3), and finally a conclusion is provided (section 3.4).

3.1 Initial approaches

The challenge we consider in this chapter is the development of an artificial vision method based on random features, which is able to determine the position of a camera device on a surface, without drawing special marks.

More precisely, the challenge is to create some methods able to find the position of an optical device over an unencoded recycled paper surface. The optical devices used in the experiments were an Epson Perfection V37 scanner using the configuration 2400x2400dpi and a PCE-MM200 Digital Microscope configured at 60x and 2400x2400dpi.

The use of direct image comparison techniques is not feasible because of the high computational cost. Thus, the initial approach was to adapt the random coding technique based on figure orientation [5], and described in section 2.2.2.2. However, this approach had to be discarded because the shape of the different features in the recycled paper were not suitable for determining their orientation, which is

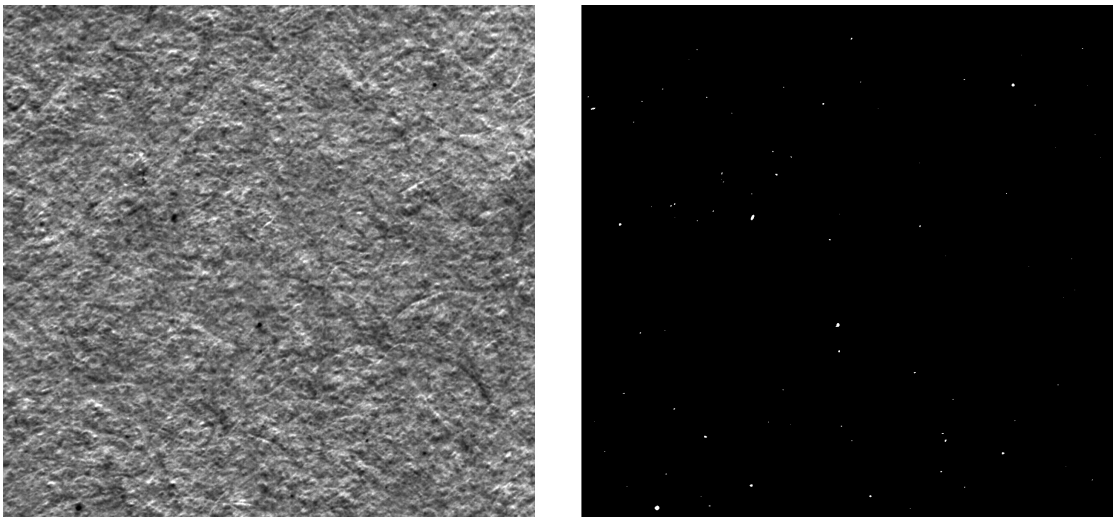

FIGURE 3.1: The left image shows the fingerprint of a recycled paper with a size of 1200x1200px and scanned with the scanner using a resolution of 2400x2400dpi. The right figure shows a binarised image, where the selected features of the left image are highlighted in white.

necessary for the technique. This occurred because the shape of the stable features on the surface of the recycled paper tend to be almost circular, and the probability of fragmentation of the few elongated features during the extraction is relatively high. Figure 3.1 shows the fingerprint of a piece of recycled paper and the extracted features from it.

The second approach focused on using SIFT [19] and SURF [11] descriptors in combination with a library for fast nearest neighbours search FLANN [15]. However, this approach was also dropped, because the computational cost and the number of spurious feature points necessary to obtain a proper correspondences was too high.

Finally, the third and successful approach was based on considering the relative position of neighbouring features in a cluster as an object and ignoring the properties of each particular feature. This allowed the storage of just the relative position information of the features on each zone of the surface, and a reasonable cost in terms of memory and computational searching time.

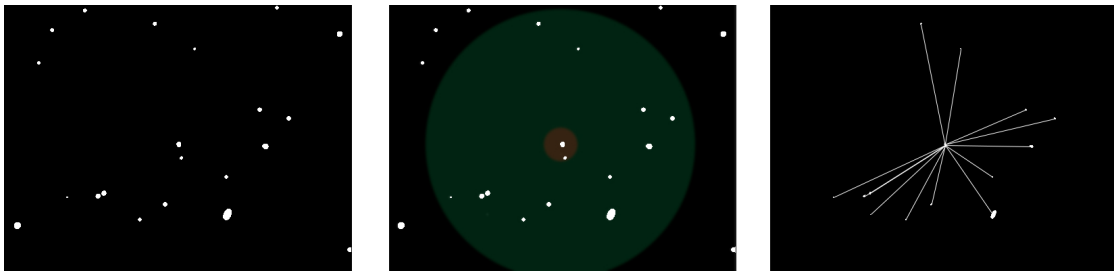

FIGURE 3.2: The left image shows the features of a fingerprint of a piece of paper. The central image displays the areas associated with the constellation of a feature, the green circular area is the area belonging to it, with the exception of the red circular area which is excluded. The right image illustrates the resulting constellation.

3.2 Theoretical estimation

The third approach is based on the assumption that the features on most of the surfaces are distributed with a uniform probability, or at least with a quasi-uniform probability. If that is the case, then the distance of the features with respect to each other must follow a cumulative Poisson distribution, and their angle to a randomly selected axis should be approximately uniformly distributed. Therefore, we can suppose that it is possible to find a random code based on vicinity of each feature

Supposing that the initial assumption is true, it is possible to define a random code associated with the relative angle between each feature and its neighbouring features. Henceforth, the descriptor of the relative positions of the group of randomly distributed neighbouring features, which surround each feature, is named a constellation. An example illustrating a constellation, extracted from the features of a recycled paper fingerprint, is shown in figure 3.2.

As shown in section 2.2.2.1, if the codes are sufficiently different, it is possible to codify each position of the surface.

3.2.1 Estimation of the information contained in a constellation

In order to estimate the number of features necessary to uniquely codify each position, we can safely fix the features on each constellation to K. Therefore, the set of angles between 0 and 360 degrees carrying information is $K - 1$. If we

estimate an experimental error in the measurements of R degrees, where $R = 2$ seems reasonable for most optical devices. Assuming N to be the number of randomly distributed points over the surface, and a uniform distribution of the angles (the distribution of angles should indeed be binomial, but assuming it to be uniform, simplifies the estimation).

Under these conditions, after matching one angle that corresponds to cN/Q points, there is a factor of $Q = c/(R+1)$ fewer possible choices, where c is a constant representing all possible rotations of the figure, in this case $c = 360$. With the second angle, the number of possible locations is cN/Q^2 . By induction, after matching $K-1$ angles the number of expected matches, due to chance, is described by equation (3.1). So, if we need to increase the area 120 times, we only need to add an extra vicinity point.

$$Matches = cN/Q^{K-1} \tag{3.1}$$

Thus, it is possible to reach a significantly small probability of finding a repeated constellation by choosing K such that $cN/Q^{K-1} << 1$. If we define the critical code length as $K_C : cN/Q^{K_C-1} = 1$, or the equivalent $K_C = \lceil \log cN / \log Q \rceil + 1$, then we can safely choose K equal to $K_C + 2$, and use several measurements, or add more criteria (such as the distance to the centre, the colour and the moments) at any feature. So for $K = 6$ and $Q = 120$ we have an estimate of $N = 120^5/360 = 69.120,000$ or linear size of order $\sqrt{N} \approx 8300$ points, which is more than sufficient for practical pointing devices or navigation systems. A line plot describing the expected number of appearances if $N = 250000$,$c = 360$ and $R = 1$ is shown in figure 3.3.

3.3 Outline of the Method

The proposed method is composed of two phases, in common with all image based recognition systems. The first one is the registering phase, where the features of the surface are extracted. A model of the whole surface is created and the information about the different constellations of the model is stored in a database or a search structure. The second phase is the positioning. In this phase the information about a certain zone of the surface is extracted and the constellations of the zone

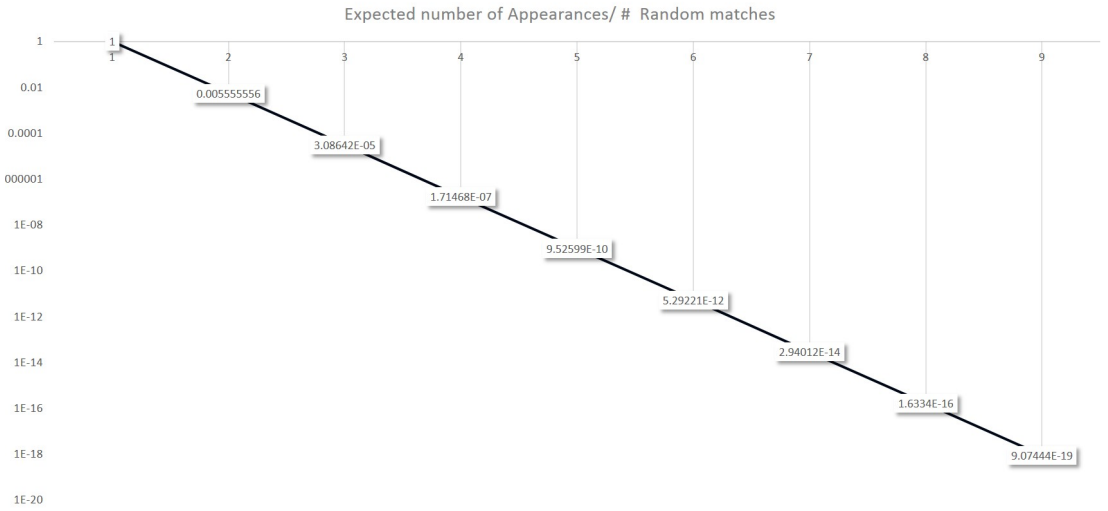

FIGURE 3.3: Graph showing the normalised relationship of the expected number of appearances and the number K of matches between two constellations, if the number N of features in the surface is 250000, the error R in the measurement of the angle is 1 and the number of possible rotations c is 360.

are compared with the constellations stored in the database (a flowchart diagram outlining the main steps is shown in figure 3.4).

3.3.1 Registering phase

The registering phase is divided into five steps.

3.3.1.1 Scanning

The first step in the registering phase is the scanning of the whole surface using an optical device. It can be performed by a direct scan of the surface, or by stitching many images to create the whole surface. However, the second option, stitching multiple images, might lead to a large positioning error due to an accumulation of distortions. There exist techniques for calibrating panned images, but these are beyond the scope of this work.

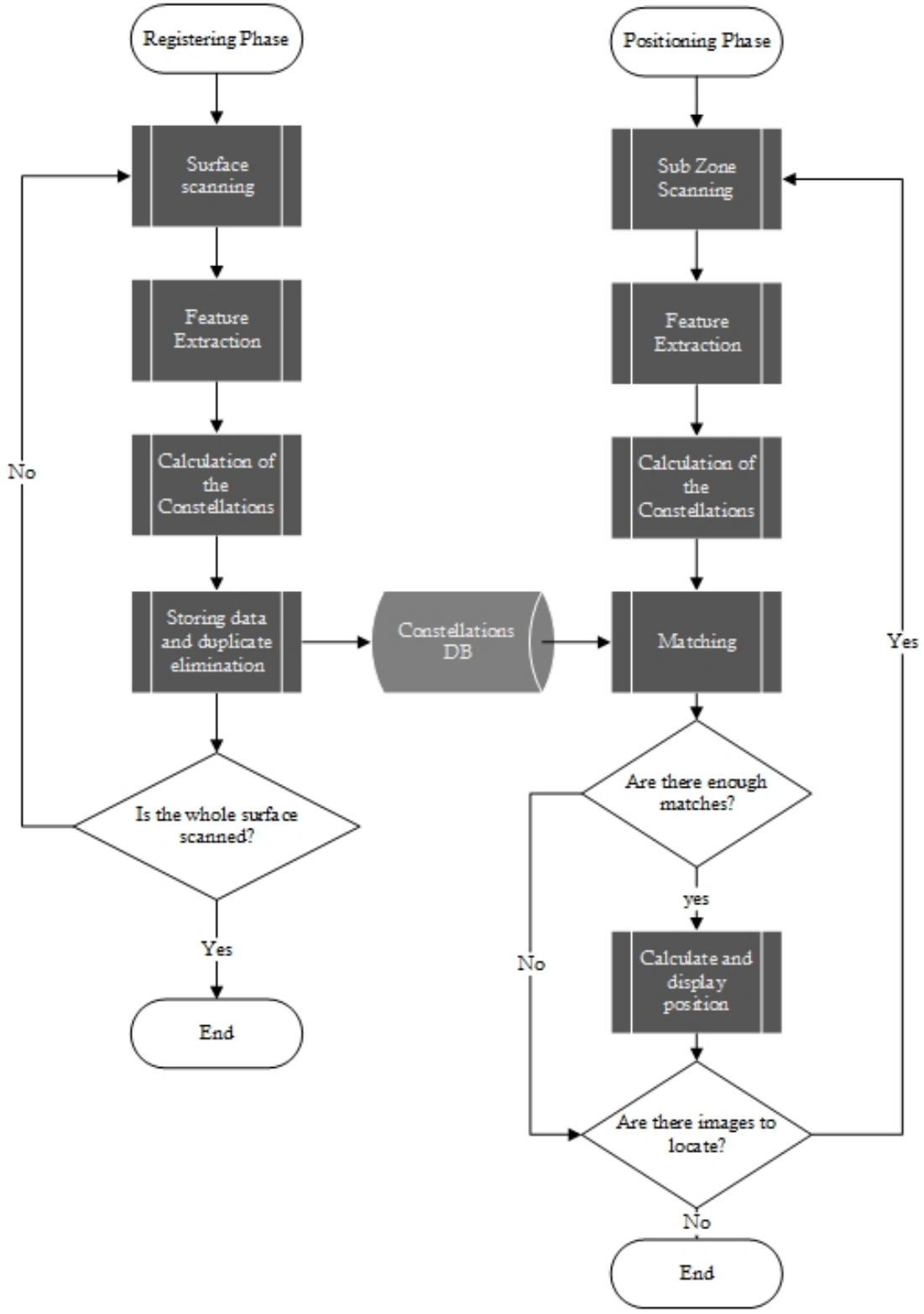

FIGURE 3.4: An outline of the phases and processes used in this method to find the position of an optical device.

3.3.1.2 Feature Extraction

The second step is to extract and describe the texture features on the surface which is necessary to identify the different zones. Some examples of the areas associated with texture features are shown in figure 3.1 and figure 3.2.

Image processing The identification of areas related to the texture features can be performed by different algorithms. These include the classic ones based on linear filters (for example Difference of Gaussians, Laplace of Gaussians, Gabor filters [47]), detection of gradients, and consequent thresholding; or by using specific techniques for a certain surface.

Feature extraction As descriptor of the features, it is mandatory to calculate the coordinates of a representative and robust point of each previously found area. We chose the coordinates of the mass centre of the features' areas. Alternatively, complex algorithms can be used to extract and describe invariant feature points such as SIFT [10], SURF [20], FAST [53], ORB [21] and BRISK [22];

Additionally, it is possible to obtain some other properties of the features, such as the spectral components, the gradients, the Hu moments[60] or the descriptors of the area. However, all these additional properties are not an integral part of the method, although they can improve its robustness and diminish the size of the area we needed to use for positioning.

3.3.1.3 Calculation of the constellations

The third step, the calculation of the constellations, comprises firstly of, selecting as a centre of the constellation each of the features. Secondly finding the neighbourhood features of the so selected constellation's centre. And thirdly, calculating the descriptor of the constellation as a set of the neighbours' feature relative position (see figure 3.2). The relative position of the neighbouring features is described by the vectors between the origin and each of the neighbours and the length of that vector, as is illustrated in figure 3.5). The axis we could choose for each constellation to measure the angles can be different, for example the angle between the shortest vector and the other the vectors, or it could be one and the same, as for example the between the X axis and the vector.

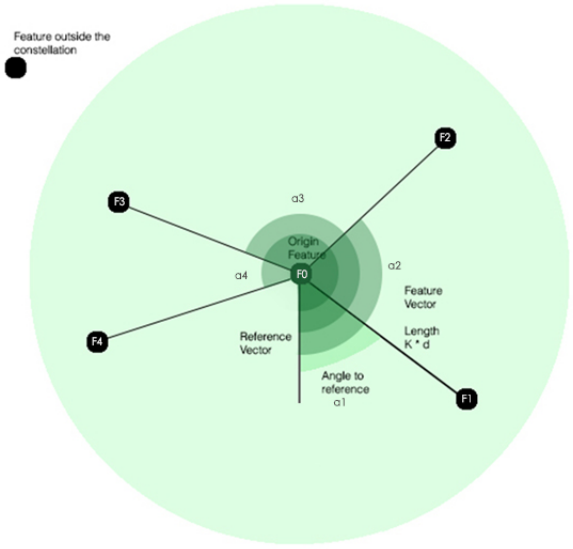

FIGURE 3.5: An illustration explaining the components of a constellation.

3.3.1.4 Storing the information

Finally, the last step in the first phase consists of storing the information about the constellations in a structure EI which allows its search. In particular, the information about the constellation C_{id} stored in EI (as shown in equation 3.2) is the id of the constellation, the chosen angle α of the axis with respect to the original reference system, the position of the centre of the constellation (x, y), the relative angles to the features of the constellation $A(i : i = 1..n)$, the length of the vectors between the neighbouring features and the central feature $L(i : i = 1..n)$, and the additional information of the features $A_g(i : i = 0..n)(i = 0$ is for the constellation's centre).

$$\forall id, C_{id} = (id, (x, y), \alpha, A(i : i = 1..n), L(i : i = 1..n), A_g(i : i = 0..n)) \in EI \tag{3.2}$$

3.3.2 Positioning phase

The Positioning phase is similar to the Registering phase but replaces the storing step by a matching step, as illustrated in figure 3.4. The phase comprises six steps: Sub-Zone Scanning, Image processing, Feature Extraction, Calculation of the Constellations, Matching, and Calculation and Displaying of Position.

3.3.2.1 Sub-Zone scanning

The first step of the phase is Sub-Zone Scanning. This obtains an image of an area of the surface by means of an optical device, with similar characteristics to the device used in the registering phase.

The Image processing, Feature Extraction, and the Calculation of the Constellation steps are identical to the steps followed in the Registering phase. However, it may be necessary to adjust the parameters to correct distortion between the images obtained by different devices in the distinct phases.

3.3.2.2 Matching step

The Matching step is performed by comparing the constellations stored in the search structure EI with each constellation of the sub-image, until two of them have enough coincidences between the relative position of their features. Further details are given in the next chapter.

3.3.2.3 Calculating and extracting the position

In the final step, the position of the image extracted by the device is calculated by applying the equation (3.3), where (x, y) are the coordinates over the surface, $(f_{x'}, f_{y'})$ the coordinates of the central feature in the image of the zone where enough matches were found, (f_x, f_y) the corresponding feature of origin in the image of the surface, (c_x, c_y) are correction constants (related to the position of the camera in the device), (w, h) are the width and the height respectively of the image obtained with the device, θ is the number of rotations of the constellation that belong to the image of the zone, necessary to obtain the maximum number of matches between both constellations, γ is a parameter to allow the transformation. Finally, the position is delivered to the user, and a new image is obtained from the device.

$$\begin{pmatrix} x \\ y \\ \gamma \end{pmatrix} = \begin{pmatrix} cos\theta & sin\theta & f_x \\ -sin\theta & cos\theta & f_y \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} w/2 + c_x - f_{x'} \\ h/2 + c_y - f_{y'} \\ 1 \end{pmatrix} \tag{3.3}$$

3.4 Conclusion

In this chapter we have discussed the possibility of using a random code, associated with the relative position between each feature and its neighbouring features, to codify each position of the surface. This is theoretically feasible as long as the features are randomly distributed on the surface. Next, the stages necessary to implement a system capable of using this random code for locating the absolute positioning of an image of the surface have been outlined.

Chapter 4

Implementation

This Chapter describes in detail the different stages of the proposed method and provides the empirical evidence of its viability. The chapter is divided into three sections. The first is a brief introduction to the technologies used for the implementation (section 4.1), the second describes in detail the surface scanning step (section 4.2), this is followed by a detailed description of the feature extraction process (section 4.3), then the steps related to the first implemented method are described 4.4 and the description of the implementation of the steps related to the second implementation 4.5. Finally, the results of the executions are provided 4.6.

4.1 Introduction to the implementation

The first implementations and tests were performed on Matlab. However, the code was migrated to C++ using the artificial vision library OpenCv [61] in order to improve the performance of the system and increase its portability.

The method was carried out in two different ways. Both methods were similar, with the exception that the steps directly related to matching, i.e. calculation of the constellations, storing data and duplicate elimination and matching differ. The first implementation of the method is based on geometric hashing, and the second one on constellation comparison.

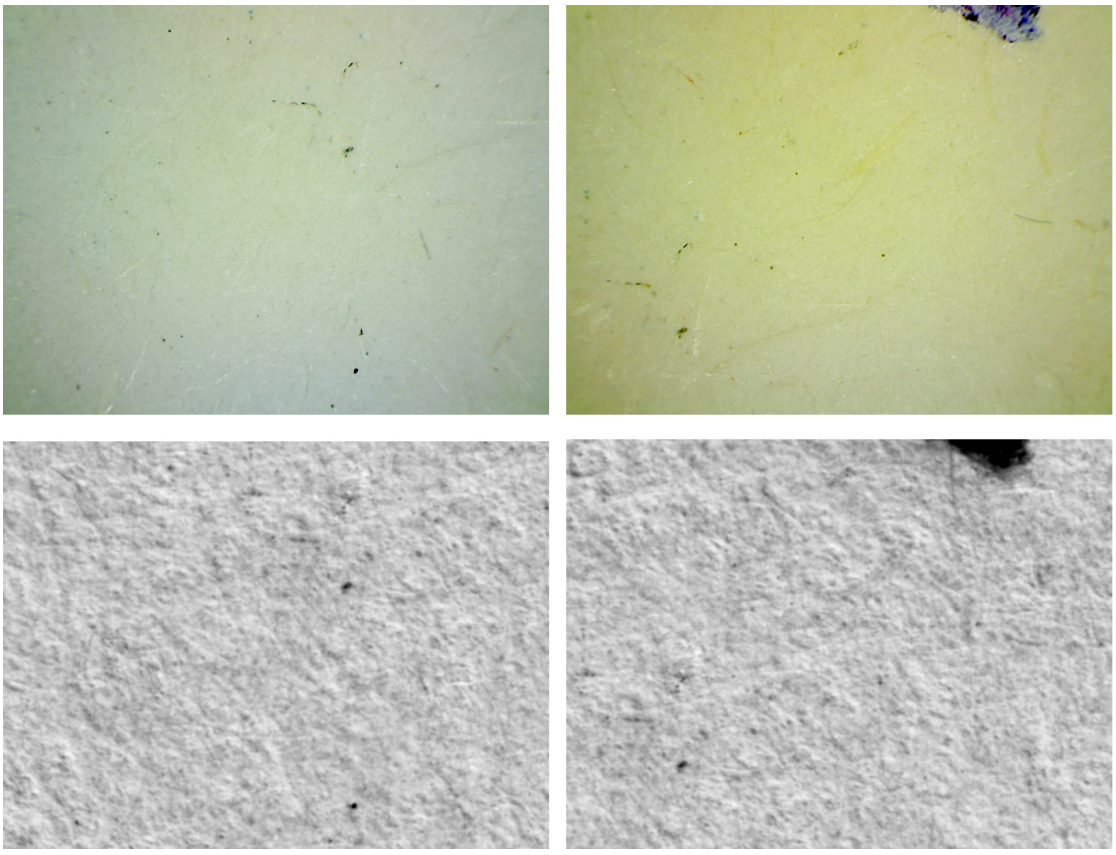

FIGURE 4.1: Images of an area of the surface of size $800x600$ px $\simeq 0.833x0.625$ cm. The top ones have been obtained using a microscope and the bottom images shows the same areas scanned with the scanner.

4.2 Surface scanning

Surface scanning is performed by means of an optical device with a resolution of 2400x2400dpi. The experiments were carried out using a Epson Perfection V37 scanner using the configuration 2400x2400dpi and a PCE-MM200 Digital Microscope configured at 60x and 0.5 Mpx. Once the image of the surface is extracted it is transformed to grayscale and tiled in overlapping regions not larger than 1400x1400 px. Figure 4.1 shows some examples of the images obtained from a sheet of recycled paper in this step.

4.3 Feature extraction

In order to properly extract the features of the surface, the different methods described in Section 2.3 were evaluated using the recycled paper texture, scanned

FIGURE 4.2: The diagram shows two images of the same surface and the SIFT features found, those corresponding to the left image are shown in red and those corresponding to the right one in green.

with the scanner. The Difference of Gaussians in combination with an adaptive threshold were found to be the methods which yielded the best reliability. When the feature extraction methods were tested with the images obtained with the microscope, which are almost noiseless, all the feature extraction methods described in Section 2.3 obtained a good reliability, once the proper parameters were chosen.

The methods based on edge detection, in particular the Prewitt and the Sobel filters, were quickly discarded due to their strong response to the edges, produced by the projection of the light used by the scanner over the irregularities of the texture.

The Gabor Filters were reliable in comparison to the other edge detection methods. However, the convolution of the bank of Gabor Filters is time consuming, consequently the Gabor Filters were also discarded.

When the SIFT and SURF algorithms were applied the key-points obtained were not as reliable as was expected from the literature. Indeed, their capability of repeatedly finding the same texture features in the images of the recycled paper was quite low, unless the number of key-points detected in a 1000x1000 px image were above 1000 points. Working with more than 1000 key-points each 1000x1000 px implies using more than 609,000 key-points per page, therefore it is not practical to use these algorithms. Also, the calculation of the key-points at the different scales is time consuming. An example of the features found by the SIFT algorithm in two different scans of the same surface is shown in figure 4.2

Blob detection methods, namely LoG and DoG described in Section 2.3.1.2, produced the most reliable results. The features found by these methods in combination with an adaptive threshold were quite robust to the different lighting

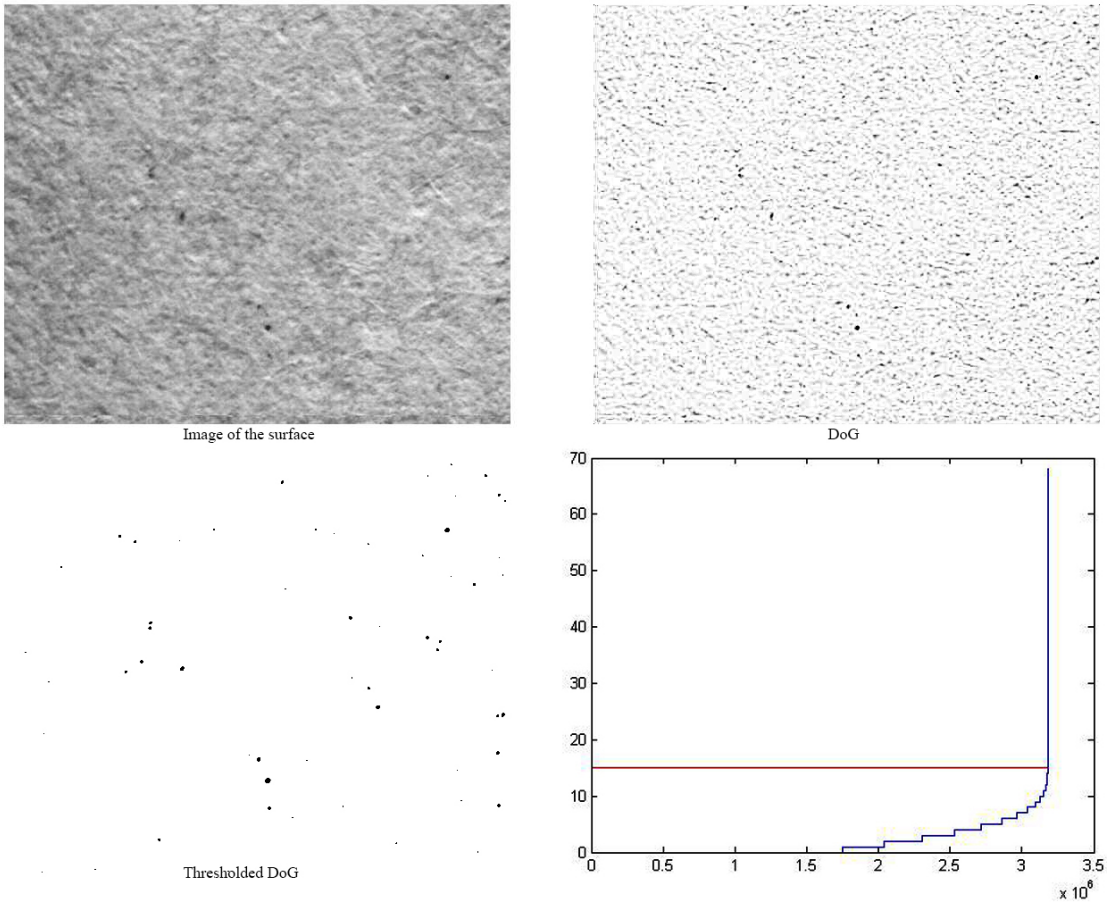

FIGURE 4.3: The top left figure displays an image of the recycled paper surface. The top right shows (with the colors inverted) the resulting image after applying the DoG filter. The bottom left image shows (with the colors inverted) the features obtained after applying the threshold to the second image. The bottom right figure plots in blue the intensity of the pixels of the top right image, sorted by their intensity value; the red line represents the threshold value associated with the 0.2 percentile of the points (used to obtain the bottom left image).

conditions.

In particular, the final implementation used a DoG filter, composed of two Gaussian Kernels with diameter 3 px and 35 px, in combination with an adaptive threshold, which is set to the value corresponding to the 0.2 percentile of the highest intensity values of the pixels belonging to the surrounding 800x800 px area.

The value of the percentile, used to set the threshold, is chosen because of the intensity of the pixels, after applying the DoG on the images of the recycled paper, can be described as a long tail distribution, and the 0.2 percentile corresponds with the highest intensity values of the distribution, as it is illustrated in figure 4.3.

$$M_x = \sum_{x,y} f(x,y) * x / \sum_{x,y} f(x,y) \tag{4.1}$$
$$M_y = \sum_{x,y} f(x,y) * y / \sum_{x,y} f(x,y)$$

In order to obtain stable coordinates of the features, the position (x,y) of the feature is obtained by calculating the mass centre (M_x, M_y), by means of the equations (**??**), of the connected components in the binarised image .

4.4 Method based on geometric hashing

The first method developed was based on geometric Hashing. As is described in section 2.4.2.2, it uses two points to fix an axis; and obtains and stores in a hash table the different models which compose the object template (in this case the triangles of the constellations), and finally finds the best matching template to the constellations of the query, using a voting system.

4.4.1 Calculation of constellations

In this implementation of the method, the calculation of the constellations is similar to the process described in section 3.3, but the axis is not fixed arbitrarily, and the constellation is decomposed in its different triangles.

In particular, the axis of the constellation is fixed in the direction of the vector formed by the central feature and its nearest feature in the constellation, with the exception of features at distances smaller than 30 px, which are ignored. The use of nearer features might lead to errors in the calculation of the angles. Then the relative angles and the distances to the other neighbouring features of the constellation are calculated.

4.4.2 Storing data and duplicate elimination

Once the triangles of the constellations are calculated, each triangle is stored in a bucket of a hash table, using as storing function the equation (4.2)

$$hashfunction = \lfloor \alpha_{id,i} \rfloor + \lfloor d_{id,i}/10 \rfloor * 360 \tag{4.2}$$

where $\alpha_{id,i}$ is the angle (in degrees) related to the feature i in the constellation id, and d_i is the distance from that feature to the central feature.

With the aim of avoiding the duplication of triangles of a constellation in the same bucket, a set structure with the id of the constellation as key is used for resolving collisions in each bucket of the hash table.

4.4.3 Matching by means of Geometric Hashing

During the matching step, each one of the constellations, obtained from the image extracted with the device, is sought in the hash map until one of them gets sufficient matches (6 should be enough if each constellation contains less than 14 features and the number of constellations stored in the hash map is smaller than 500,000).

In order to search for the constellation in the hash map, each triangle of the constellation (α_i, d_i) is searched for in the Hash Map, its associate buckets are found, the votes are organised in an accumulator, and the id of the bucket with the largest number of votes is returned. This process is illustrated in figure 4.4.

4.5 Method based on bit-masks

The method based on geometric hashing has the advantage of being extremely fast ($O(t * f)$, where t is the average number of triangles in the buckets of the hash table and f is the average number of features in the constellations). However, the probability of not finding a match is high when the images are noisy, because the feature used to form the axis might not be found; also, the memory requirements needed to minimise the effects of this problem are high $O(m * f * (R + 1))$, R

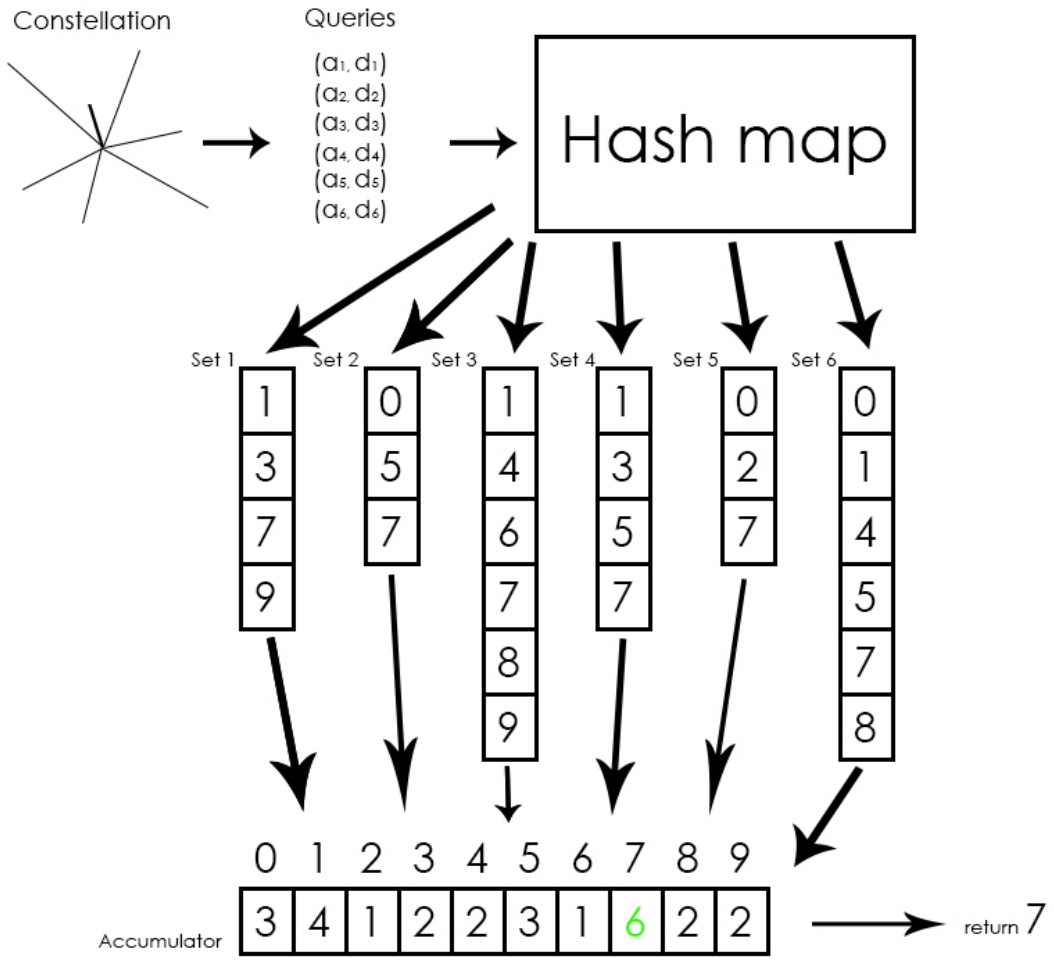

FIGURE 4.4: Illustration describing the process followed in the matching step of the implementation based on geometric hashing.

being the expected error in the measurement of the angle, and m the number of constellations on the whole scanned surface.

Thus, a second method capable of performing an intensive search of the constellations, with a low memory cost $O(m)$ and a reasonable execution time $O(m)$, was developed. It is important to note that this method is easy parallelizable on GPUs or FPGAs. It uses two rotatory bit-masks to compare the constellations of the image and the models.

4.5.1 Calculation of constellations

In this method the calculation of the constellations is identical to the process described in section 3.3. However, the neighbouring features at a distance smaller

than 30 px are discarded, and in addition to the calculation of the relative position of each element of the constellation, two bit-masks are built, one representing the relative angle to each neighbouring feature, and the other the distance from the central feature to each neighbouring feature.

The angles bit-mask is obtained by setting to '1' each bit b_{α_i} of the mask whose index corresponds to the angle α_i, where α_i corresponds to the angle to the axis of the neighbouring feature i in the constellation; the remaining bits are set to '0'.

The distances bit-mask is generated by setting to '1' each bit b_{d_i} of the mask whose index corresponds to $\lfloor d_i/2M \rfloor$, where the distance d_i corresponds to the distance from the neighbouring feature i to the central feature of the constellation, and M represents the maximum expected error in the measurements; the remaining bits are set to '0'.

4.5.2 Storing data and duplicate elimination

Next, the angles masks are fattened before being stored, to minimise the errors produced by distortions in the measurements. The fattening is performed by setting the b_i adjacent bits to b_{α_i} to '1', being b_i a value given by the equation (4.3).

$$b_i = \arctan(M/d_i) * 360/2\pi; \tag{4.3}$$

After fattening the masks of the constellations, the two masks and the information of each constellation are stored in the rows of a table, avoiding duplicates.

4.5.3 Matching by means of bit-masks representing polar coordinates

In this implementation the matching step is performed by comparing each possible rotation of the angles-mask of a constellation extracted with the device and the corresponding distance mask, with the masks stored in the table. If the number of coincidences between the masks is higher than 5 then the relative positions of the constellations are checked and if the number of real matches is higher than 6 the match is provided. It is recommended that not more than 5 matches are

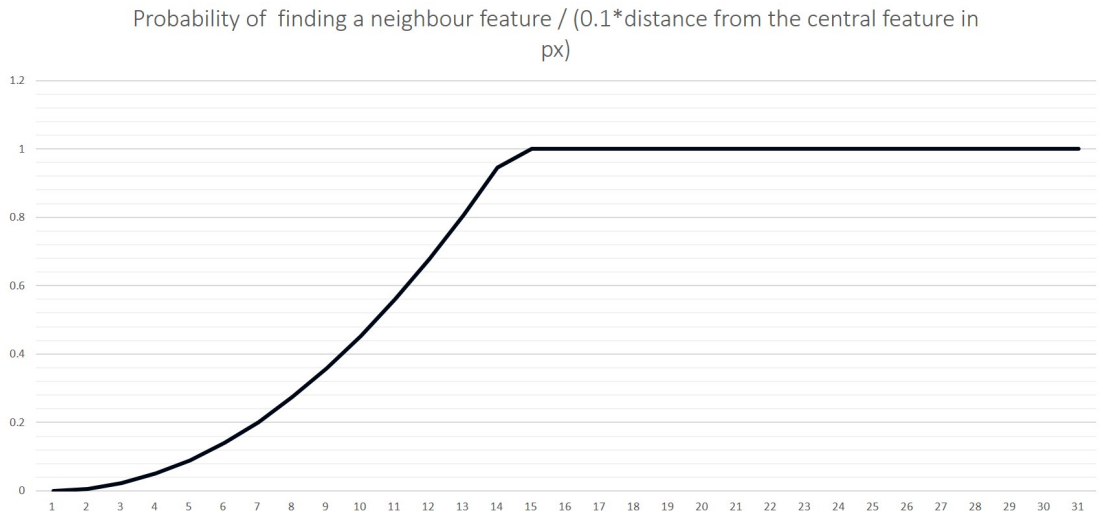

FIGURE 4.5: Distribution of the probability of finding a feature at distance d in a din-A4 surface with 10,000 features on it. Each unit in the x axis represents a distance of 10 pixels on the image of the surface.

used when the masks are compared, at least with the distance mask, since the probability of finding a feature at distance d follows a cumulative distribution, and 5 'distance bits' may correspond to all the features in the constellation; the probability distribution of finding a feature at distance d is shown in figure 4.5 .

4.6 Experimental results

4.6.1 Erroneous matches

The first experiment is focused on finding out how the number of erroneous appearances evolves with the number of matched triangles. To do so, nineteen din-A4 recycled paper sheets, with a total of 253218 feature points, were compared against 35 samples, of size 1400x1400 px and an average of 52 features per sample, extracted from a different recycled sheet of paper. The results are plotted in figure 4.6.

Figure 4.6 shows a continuous line with the distribution of matches obtained and a dashed line with the theoretical distribution of random matches described by equation (3.1).

As expected, the number of matches obtained follows a exponential distribution, similar to the theoretical estimation. However, the number of matches with just

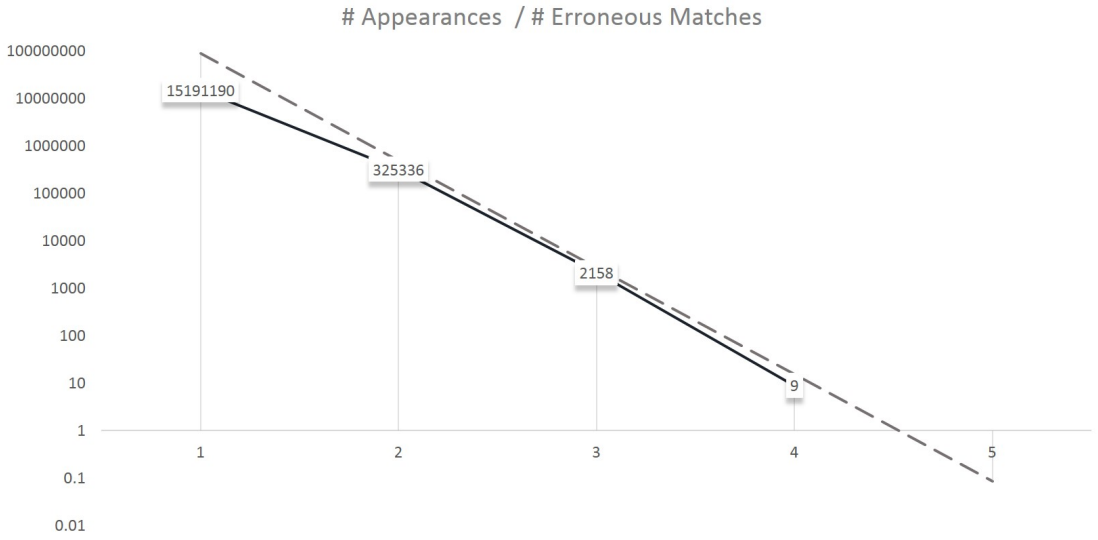

FIGURE 4.6: Graph showing the relationship between the number of appearances of m erroneous matches between two constellations. The test was performed with 35 random samples of size 1400x1400 px, extracted with the scanner from a din-A4 recycled paper sheet containing 12682 features; against other nineteen din-A4 recycled paper sheets with a total of 253218 features. The dashed line represents the expected value given by equation (2.4).

one triangle is significantly smaller than the expected value. This discordance is produced because the implementation of the method uses not only the angles, but also the distances to the central feature for the matching step.

Since the distances distribution accumulates most of the points in the constellations in the buckets corresponding to the range [150,200] px (i.e. five buckets), the number of matches found with more than two coincidences is not strongly conditioned by this additional restriction, but it has a high impact when there is just one match.

We have not found more than 6 erroneous matches between constellations in any of the experiments.

4.6.2 Executions

Now, the objective is to test the performance and the reliability of the two proposed implementations of the method. The first using a geometric hash structure to match the constellations and the second performing the matching step through bit-masks.

The tests were performed using 100 random samples from the sheet 1a, extracted using the scanner. The area of each sample is 1400x1400 px. Then the position of each sample is located in different scans of the sheet: without rotation (1a+0°), without rotation and poor lighting conditions (1a+Poor lighting), with -10 degrees of rotation (1a-10°), with 168 degrees of rotation (1a+168°), with 180 degrees of rotation (1a+180°), and a different sheet (1b). The rotation of the sheets during the scanning is critical because the light of the scanner is not perpendicular to the surface. Table 4.1 shows the results of the executions.

Successful positioning on sheet 1a using a scanner

Sheet	Successful positioning with hash	Median retrieval time using hash (ms)	Successful positioning with masks	Median retrieval time using masks (ms)
1a vs 1a	100 %	3	100 %	265
1a vs 1a+0°	100 %	3	100 %	308
1a vs 1a-10°	93 %	4	93 %	343
1a vs 1a+168°	23 %	12	71 %	3591
1a vs 1a+180°	17 %	18	90 %	2911
1a vs 1a+Poor lighting	86 %	8	100 %	2600
1a vs 1b	0 %	24	0 %	132821

TABLE 4.1: Success rate in positioning and retrieval time for both implementations of the method.

In table 4.1 we observe that the speed of the method based on Hashing clearly outperforms the execution performance obtained with the rotation of the bit-masks. However, in terms of reliability the bit-masks method performs much better when there are variations in the lighting conditions.

We can see that both methods have 100 % reliability when the lighting conditions are almost identical to the test scan (1a). Also, when the sheet is rotated 10 degrees (1a-10°) the reliability is almost 100 %. Actually, the result obtained is 93 % but it is necessary to take into account that because of the rotation 8 % of the page is out of the scanned image; thus, those areas cannot be located.

Even when the lighting conditions are different (1a+Poor lighting), the performance is quite good for both algorithms 86 % success using hash and 100 % using the bit-masks.

When the lighting conditions radically change, as happened with the images 1a+180° and 1a+168°, where the light strikes in the opposite direction, the results

are worse. A 17 % and 23 % probability of success are obtained when the hash was applied, and 90 % and 71 % when the bit-masks were used. The 17 % detection may seem small for practical use. But bearing in mind that we need only one good match to identify the position and that the probability of error is almost 0 % we can use the scans of the surrounding areas to locate the position. As expected there were no coincidences with the sheet 1b.

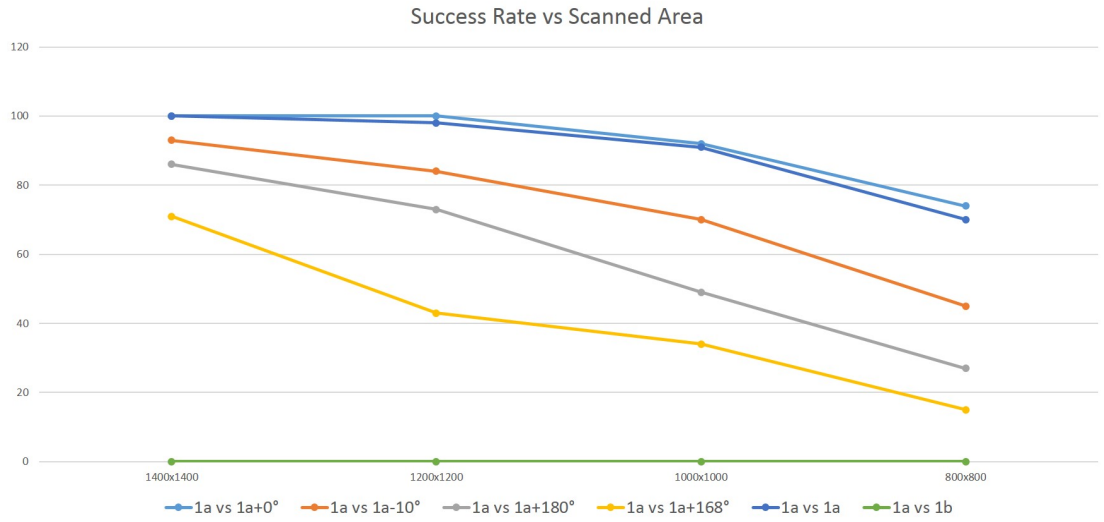

FIGURE 4.7: Evolution of the success rate as function of the test area.

To analyse how the size of the test area affects the results when the images are extracted with the scanner, the experiment was repeated using different test area sizes. As expected, the number of matches decreases with the area, as is shown in figure 4.7.

Also, the implementations of the method were tested using the images obtained using the microscope. A group of 100 images were obtained, where each one is overlapped with at least the previous and the next image.

The results are shown in table 4.2. We can observe a good performance for both algorithms. Surprisingly, the number of successes of the algorithms is much better than the performance obtained with an area of 800x800 px when the image was extracted using the scanner, and the image 1a of the scanned area is compared against itself. This happens because the features in the images obtained using the microscope are more difficult to confuse with the noise of the background, and the images extracted using the scanner have many features that are ignored to avoid the selection of the noise as a feature.

Successful detection of overlapping regions

Successful positioning with masks	Median retrieval time using masks (ms)	Successful positioning with hash	Median retrieval time using hash (ms)
98 %	25	92 %	3

TABLE 4.2: Success rate in the positioning and retrieval time of the two implementations of the method, using 100 overlapped images of size 800x600 px of a recycled paper surface obtained using a microscope.

4.7 Conclusion

In this chapter we have discussed the implementation details of the different stages (surface scanning, feature extraction, calculation of the constellations, storing data and duplicate elimination, and matching) of the two implementations of the method.

Finally, the empirical results of the executions of both implementations have been analysed. As a result we have a 100 % success rate finding the position when most of the features are properly extracted, which validates the method. However, this result is strongly dependent on the reliability of the extracted features. Thus the number of successfully retrieved positions decreases when the lighting and optical conditions make the feature extraction process, difficult.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Absolute positioning is a complex problem with multiple challenges which are yet still unresolved. In this work we propose a new approach for facing one of these unresolved challenges, the location of optical devices over surfaces without modifying them; when the surfaces have randomly or quasi-randomly distributed features, and the features can be reliably recognised by optical means.

After discarding multiple approaches cited in the literature used for positioning on printed surfaces and for object recognition, which are not suitable for the recognition of positions on random surfaces, due to the similarities between the features. We found that the relative positions of neighbouring features used as a random code can determine the location on the surface with a low probability of error. A probability of error that can be adjusted by increasing the number of required matches in the neighbours' vicinity.

The importance of extracting reliable features for the code, necessitated the testing of different feature extraction methods. The combination of the DoG filter with an adaptive threshold, performed best in our experiments on recycled paper.

We developed two implementations of the method to test the performance and the reliability of the code at finding the absolute position. The first implementation, based on geometric hashing had a good execution time, but its reliability was poor

when the lighting conditions changed. The second implementation, based on bit-masks comparison was reliable even with low lighting conditions, but had a slow execution time.

The empirical results of both implementations show a reasonable balance between reliability, execution time, and memory. The results were coherent with the theoretical estimations of the probability of error, validating the proposed method for absolute positioning on recycled paper surfaces. In addition, this is a method which can be easily adapted to other surfaces, as long as we can optically find randomly distributed features on them.

As a result, we have found a method which uses optically extracted data to locate the absolute position of a device, based on the irregularities of the texture of a surface. The method avoids direct image comparison, and is sufficiently fast to be used in a pointer device.

5.2 Future Work

This method allows the development of multiple industrial applications, some of which are:

- The detection of uniqueness and the reconstruction of artwork from their texture.
- The development of biometric devices able to determine its position on the skin.
- The construction of indoor navigating robots able to precisely determine and share their position, without external support.
- The development of virtual interfaces based on pointer devices.
- The substitution of Anoto digital paper by random surfaces.
- The construction of a cleaner robot able to determine its position on solar panels.

On the other hand, from a research point of view there are many aspects of the method which could be improved, such as:

- The improvement of the feature detection algorithms under difficult lighting conditions.
- The detection of feature points at multiple scales.
- The automatic selection of the feature extraction method and parameters as a function of the texture.
- The combination of the two implemented methods to improve the average speed and reliability of the method.
- The implementation of the method using parallel architectures such as FPGAs or GPUs
- The generalization of the method to three dimensional spaces.

Bibliography

[1] Mats Petter Pettersson and Tomas Edso. Coding pattern and apparatus and method for determining a value of at least one mark of a coding pattern, December 16 2003. US Patent 6,663,008.

[2] De Bruijn binary sequence of size 4. `http://en.wikipedia.org/wiki/De_Bruijn_sequence#mediaviewer/File:De_Bruijn_sequence.svg`. Accessed: 2014-08-25.

[3] Edward Aboufadel, Timothy Armstrong, and Elizabeth Smietana. Position coding. *arXiv preprint arXiv:0706.0869*, 2007.

[4] Kamen Kanev and Shigeo Kimura. Clustering-scheme-encoded interfaces providing orientation feedback, August 2 2011. US Patent 7,991,191.

[5] Kostadin Koroutchev and Elka Korutcheva. Figures design for surface coding with orientation. 2011.

[6] Xudong Xie, Kin-Man Lam, Hongya Zhao, and Qionghai Dai. Efficient rotation-and scale-invariant texture classification method based on gabor wavelets. *Journal of Electronic Imaging*, 17(4):043026–043026, 2008.

[7] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[8] Francis HY Chan, Francis K Lam, and Hui Zhu. Adaptive thresholding by variational method. *IEEE Transactions on Image Processing*, 7(3):468–473, 1998.

[9] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30 (1):32–46, 1985.

[10] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[11] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

[12] Simon Moss, Richard C Wilson, and Edwin R Hancock. A mixture model for pose clustering. *Pattern Recognition Letters*, 20(11):1093–1101, 1999.

[13] Yehezkel Lamdan and Haim J Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *ICCV*, volume 88, pages 238–249, 1988.

[14] Thomas F Knight and Eric Nestler. Optical mouse, January 17 1989. US Patent 4,799,055.

[15] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP (1)*, pages 331–340, 2009.

[16] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):881–892, 2002.

[17] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[18] Takahiko Furuya and Ryutarou Ohbuchi. Dense sampling and fast encoding for 3d model retrieval using bag-of-visual features. In *Proceedings of the ACM international conference on image and video retrieval*, page 26. ACM, 2009.

[19] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. IEEE, 1999.

[20] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3): 346–359, 2008.

[21] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[22] Stefan Leutenegger, Margarita Chli, and Roland Yves Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555. IEEE, 2011.

[23] George Stockman. Object recognition and localization via pose clustering. *Computer Vision, Graphics, and Image Processing*, 40(3):361–387, 1987.

[24] Clark F Olson. Efficient pose clustering using a randomized algorithm. *International Journal of Computer Vision*, 23(2):131–147, 1997.

[25] Mats Petter Pettersson and Tomas Edso. Determination of a position code, April 15 2003. US Patent 6,548,768.

[26] Christer Fåhraeus, Johan Lindgren, and Stefan Burström. Electronic pen, July 3 2007. US Patent 7,239,306.

[27] M.P. Pettersson and T. Edsoe. Encoded paper for optical reading, April 12 2001. URL `http://www.google.es/patents/WO2001026032A1?cl=en`. WO Patent App. PCT/SE2000/001,895.

[28] Petter Ericson and Stefan Burström. Systems and methods for printing by using a position-coding pattern, February 15 2005. US Patent 6,854,821.

[29] Emiliano Bartolome, Manuel Gonzalez, and Andreu Gonzalez. Data encoding pattern, January 24 2012. US Patent 8,100,338.

[30] Oral F Sekendur. Absolute optical position determination, December 22 1998. US Patent 5,852,434.

[31] S.R. Borgaonkar and P. Dey. A pointing device with absolute and relative positioning capability, December 28 2006. URL `http://www.google.com/patents/WO2006137077A1?cl=en`. WO Patent App. PCT/IN2005/000,209.

[32] Alison K Brown and Mark A Sturza. Gps tracking system, January 3 1995. US Patent 5,379,224.

[33] Anthony J Weiss. Direct position determination of narrowband radio frequency transmitters. *Signal Processing Letters, IEEE*, 11(5):513–516, 2004.

[34] Yoko Sasaki, Satoshi Kagami, and Hiroshi Mizoguchi. Multiple sound source mapping for a mobile robot by self-motion triangulation. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 380–385. IEEE, 2006.

[35] Seyed Ehsan Marjani Bajestani and Arsham Vosoughinia. Technical report of building a line follower robot. In *Electronics and Information Engineering (ICEIE), 2010 International Conference On*, volume 1, pages V1–1. IEEE, 2010.

[36] Jun Ota, Masakazu Yamamoto, Kazuo Ikeda, Yasumichi Aiyama, and Tamio Arai. Environmental support method for mobile robots using visual marks with memory storage. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 4, pages 2976–2981. IEEE, 1999.

[37] Arnold Irschara, Viktor Kaufmann, Manfred Klopschitz, Horst Bischof, and Franz Leberl. *Towards fully automatic photogrammetric reconstruction using digital images taken from UAVs*. na, 2010.

[38] Felipe Pinage, Jose Reginaldo Hughes Carvalho, Emory Raphael Viana Freitas, and Jose Pinheiro de Queiroz Neto. Feature transform technique for combining landmark detection and tracking of visual information of large rain forest areas. In *Robotics Symposium and Competition (LARS/LARC), 2013 Latin American*, pages 30–37. IEEE, 2013.

[39] TW Ng. The optical mouse as a two-dimensional displacement sensor. *Sensors and Actuators A: Physical*, 107(1):21–25, 2003.

[40] Kamen Kanev and Shigeo Kimura. Digital information carrier, 2005. JP Patent No 3,635,374.

[41] Stefan Lynggaard and Mats Petter Pettersson. Devices method and computer program for position determination, November 22 2005. US Patent 6,966,495.

[42] C. Fåhraeus, J. Lindgren, and S. Burström. Electronic pen, July 3 2007. URL `http://www.google.com/patents/US7239306`. US Patent 7,239,306.

[43] Nicolaas Govert de Bruijn and Paul Erdos. A combinatorial problem. *Koninklijke Nederlandse Akademie v. Wetenschappen*, 49(49):758–764, 1946.

[44] Judith MS Prewitt. Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1):15–19, 1970.

[45] S Marčelja. Mathematical description of the responses of simple cortical cells*. *JOSA*, 70(11):1297–1300, 1980.

[46] Judson P Jones and Larry A Palmer. An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex. *Journal of neurophysiology*, 58(6):1233–1258, 1987.

[47] Francesco Bianconi and Antonio Fernández. Evaluation of the effects of gabor filter parameters on texture classification. *Pattern Recognition*, 40(12):3325–3335, 2007.

[48] Anil K Jain and Farshid Farrokhnia. Unsupervised texture segmentation using gabor filters. In *Systems, Man and Cybernetics, 1990. Conference Proceedings., IEEE International Conference on*, pages 14–19. IEEE, 1990.

[49] Lars Bretzner and Tony Lindeberg. Feature tracking with automatic selection of spatial scales. *Computer Vision and Image Understanding*, 71(3):385–392, 1998.

[50] Richard A Young. The gaussian derivative model for spatial vision: I. retinal mechanisms. *Spatial vision*, 2(4):273–293, 1987.

[51] L Shapiro and G. Stockman. *Computer Vision*, chapter 3. Prentice Hall, Upper Saddle River, N.J., 2002.

[52] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517. IEEE, 2012.

[53] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006*, pages 430–443. Springer, 2006.

[54] Luo Juan and Oubong Gwun. A comparison of sift, pca-sift and surf. *International Journal of Image Processing (IJIP)*, 3(4):143–152, 2009.

[55] Konstantinos G Derpanis. The gaussian pyramid. 2005.

[56] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[57] Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 525–531. IEEE, 2001.

[58] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[59] Jacob T Schwartz and Micha Sharir. Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves. *The International Journal of Robotics Research*, 6(2):29–44, 1987.

[60] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962.

[61] Gary Bradski. The opencv library. *Doctor Dobbs Journal*, 25(11):120–126, 2000.

Appendix A

Publications

This work has been presented as technological proposal by FUAM and nowadays it is in state of Patent Pending with number 201431308.